(2) LEVEL II

RADC-TR-79-206
Final Technical Report
August 1979

TOPS 20 NSW

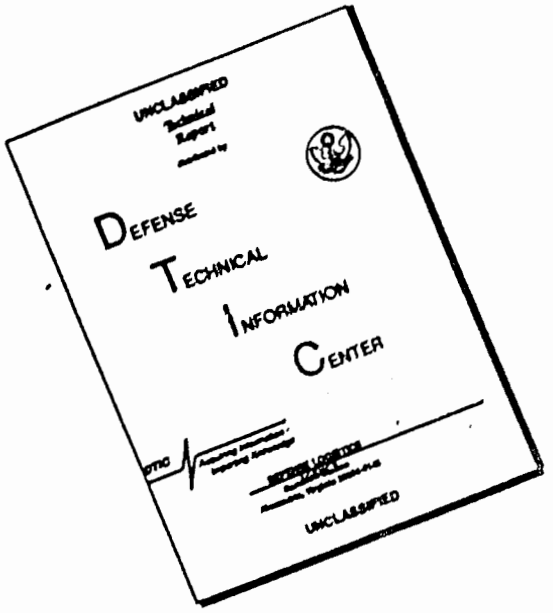
Massachusetts Computer Associates

Sponsored by
Defense Advanced Research Projects Agency (DoD)
ARPA Order No. 3686

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-79-206 has been reviewed and is approved for publication.

APPROVED: Richard A Robinson

RICHARD A. ROBINSON
Project Engineer

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISCP), Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

TOPS 20 NSW

Charles A. Muntz

Contractor: Massachusetts Computer Associates
Contract Number: F30602-78-C-0087
Effective Date of Contract: 1 October 1977
Contract Expiration Date: 6 December 1978
Short Title of Work: TOPS 20 NSW
Program Code Number: 8P10
Period of Work Covered: Oct 77 - Dec 78

Principal Investigator: Charles Muntz
Phone: 617 245-9540

Project Engineer: Richard A. Robinson
Phone: 315 330-7746

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by Richard A. Robinson (ISCP), Griffiss AFB NY 13441 under Contract F30602-78-C-0087.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (18) RADC-TR-79-206	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) TOPS 20 NSW		5. TYPE OF REPORT & PERIOD COVERED (9) Final Technical Report Oct 77 – Dec 78
		6. PERFORMING ORG. REPORT NUMBER (14) CADD-7901-0511
7. AUTHOR(s) (10) Charles A. Muntz		8. CONTRACT OR GRANT NUMBER(s) (15) F30602-78-C-0087 ARPA Order-3686
9. PERFORMING ORGANIZATION NAME AND ADDRESS Massachusetts Computer Associates 26 Princess Street Wakefield MA 01880 093 745		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62708F (16) 25310[illegible] (17) 01
11. CONTROLLING OFFICE NAME AND ADDRESS (11) Defense Advanced Research Projects Agency 1400 Wilson Blvd Arlington VA 22209		12. REPORT DATE Aug 79
		13. NUMBER OF PAGES 89
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Rome Air Development Center (ISCP) Griffiss AFB NY 13441		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.
(12) 88

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
Same

18. SUPPLEMENTARY NOTES
RADC Project Engineer: Richard A. Robinson (ISCP)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Software Systems
Software Engineering
Computer Networks

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
The National Software Works (NSW) is a distributed software system resident on the ARPANET. It is intended to support the development, use, maintenance, and storage of programs and data on which the programs operate. It is principally aimed at the development of software systems and at providing software tools which can be used to support the software development activity throughout its life cycle.

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

093 745

TABLE OF CONTENTS

1. Introduction

The National Software Works (NSW) is a significant new step in the development of distributed processing systems and computer networks. NSW is an ambitious project to link a set of geographically distributed and diverse hosts with an operating system which appears as a single entity to a prospective user.

The National Software Works is being developed in response to a growing concern over the high cost of software. The Air Force has estimated that in FY72 it spent between $1 billion and $1.5 billion on software, about three times the annual expenditure on computer hardware. The Air Force has further estimated that by 1985 software expenditures will be over 90% of total computer system costs.

Since the early days of computing, in fact, the cost and complexity of developing and maintaining software have been substantial obstacles to the efficient and effective use of computers. To breach this barrier, both industry and government have committed vast resources for the development of tools -- automated aids for the implementors of software and the managers of software implementation projects. These tools include compilers, editors, debuggers, design systems, test management tools, language analyzers, etc.

The difficulty is not the existence of suitable tools for a given programming task; it is the availability of the tools. The notion of software portability, often proposed as the solution for the problem of providing programming tools in some environment, has proven to be a will-o'-the-wisp which the industry has vainly pursued for the past twenty years.

The success of the Arpanet in providing programmers economical access to geographically dispersed computers provided the foundation on which the NSW concept was built. Instead of moving the software from host to host, let the programmer (and manager) use each software tool on whatever host it already occupies. To take a specific example, the Navy requires a programming support environment for the UYK-20 minicomputer. There currently exist cross-assemblers and compilers for the UYK-20 on IBM 360 hardware. On TENEX there is a UYK-20 emulator and debugger. MULTICS has the QEDX editor. All three of these host computers are connected by the Arpanet. Solution -- let the programmer use these existing tools to develop UYK-20 software.

That solution sounds plausible, but it ignores some serious practical considerations.

- o You need an account on each host. This involves the allocation of funding, drawing up contracts, etc.

- o The operating system on each host is different, so you must learn different login procedures, command languages, interrupt characters, file naming conventions, etc. Further you must not confuse each system's conventions as you move from tool to tool.

- o Files output from one tool (say QEDX on MULTICS) are to be input to another tool (say CMS2M on IBM 360). This involves at least network transmission and usually file reformatting. To appreciate the magnitude of this problem one should try to use FTP (Arpanet File Transfer Protocol) to move a QEDX output file -- a sequential file of 9 bit ASCII characters in 36 bit words -- to an IBM 360 to be a CMS2M input file -- a blocked file of 80 EBCDIC character records in 32 bit words.

These and similar problems will be familiar to anyone who has used several different systems.

The purpose of NSW is to make this solution (of providing programmers access to tools on different hosts) a practical reality. The NSW user should not have to know about OS/360, TENEX, and MULTICS with their differing file systems, login procedures, system commands, etc.; knowledge of how to use the individual tools which are needed for the job should suffice. He should not have to worry about reformatting and moving files from a 360 to a TENEX; file transmission should be completely transparent. The user should not have to worry about obtaining accounts on many different machines, but instead should have a single NSW account.

Thus, the National Software Works is to provide programmers with a

- o Unified tool kit - distributed over many hosts, and a

- o Single monitor with
  - . uniform command language,
  - . global file system,
  - . single access control, accounting, and auditing mechanism.

2. History of NSW Project

2.1 NSW Goals

As originally conceived, NSW was to provide the above-described facility in the context of certain specific external goals. The first such goal was large scale. Contemporary operating systems support tens of concurrent users. NSW was to support many more users, possibly as many as one thousand. The catalogue alone of the file system for that many users could easily fill a 3330 disk pack. The table space required for keeping track of one thousand users and the software tools that they are using could easily exceed the memory of a medium size TENEX.

The second goal was high reliability. If there are one thousand online users, then a two hour system failure costs one man-year of work. The National Software Works -- particularly its monitor and file system -- must degrade gracefully. Failure of a single component -- e.g., a TENEX system on which tools are running -- must only reduce system capacity, not destroy it. Further, only those users actually using a failed component should be affected by its failure.

The third goal was support of project management. NSW was to provide managers of software projects with a collection of programs, called management tools, which they can use to monitor and control project activities. The underlying assumption here is that a manager's ability to insure that each programmer's efforts contribute most effectively to overall project goals can be greatly enhanced by automating routine management tasks. Furthermore, it is assumed that a good environment for this automation is the system which supports the project programming activities because it represents an effective point for monitoring and controlling those activities.

The fourth goal of NSW was practicality. NSW was not to be a "blue sky" system, whose implementation required unrealistic assumptions about its environment. In particular, practicality meant:

- Minimum modifications to existing operating systems on Arpanet hosts. Minimum was, in fact, to be construed as none. It was possible to add privileged (i.e., non-user) code to the existing systems, but the solution to the problem should not depend on rewriting the kernel of any existing operating system.

- Minimum modifications to existing tools. Here, minimum no longer meant none. It was possible to require some change to a tool as part of the process of NSW installation, but such changes should be small scale and contained.

- Maximum generality. Any solution which permits the easy installation of existing tools must also allow the easy construction and installation of new tools.

- o No experimental hardware. This requirement meant that new hardware-oriented-approaches to reliability - e.g., PLURIBUS - cannot be used. The NSW monitor and file system are to run on already available Arpanet hosts.

2.2 NSW Architecture

In this section, we summarize the NSW design, indicating what effect the NSW goals had on the system architecture. We can factor the NSW problem into two parts:

- o The development and implementation of methodology for excising tools from their current environments and interfacing them with the new NSW monitor.
- o The design and construction of a unified monitor and file system for the Arpanet.

In the next two subsections we examine each of these problems in turn and describe the components of NSW which provide solutions to the technical difficulties involved with each part.

2.2.1 Tool Kit

We first have the task of excising tools from their current operating environment and embedding them in the new one. In the context of the goals of NSW, we will discuss the technical issues which must be solved in order to provide the requisite tool installation methodology.

By its very definition, NSW is a distributed system. Tool processes run on different Arpanet hosts. The monitor must run on at least one Arpanet host. There must be some form of inter-host inter-process communication. There are low level Arpanet protocols for moving bits from host to host, and there are also several higher level protocols for moving files and for terminal communication. None of these protocols, however, is oriented toward the kind of inter-process communication which NSW requires. Moreover, even though NSW is being implemented on the Arpanet, we want to keep it as independent as possible of the underlying milieu. Network technology is evolving, and we wish to be able to realize the NSW architecture on tomorrows's network's as well. Hence, the first technical problem to be solved is the definition and implementation of an appropriate inter-host inter-process communication protocol. The protocol developed for NSW is called MSG.

The user of a tool has a variety of mechanisms for communicating with the tool. The user's terminal must be interfaced to the system and its peculiarities handled -- for example, the right amount of padding added after a carriage return. Control characters which happen to be meaningful to the local host must be intercepted before they reach the local executive. In order to allow uniform access to all the tools in NSW, running on many different machines, we must define a standard set of control functions and implement a system component which interfaces the user to every tool. The problem of standardizing control functions and insulating the user from the vagaries of the different operating systems is handled by an NSW component called the Front End.

A tool running on some machine makes system calls requesting resources -- primarily file access. Since access to NSW system resources is to be controlled, accounted for, and audited by the NSW monitor, such requests must be diverted from the local system and instead referred to that monitor. In addition, the tool expects to have a communication link with the user, and this link in NSW is via MSG to the Front End. So, without modifying the operating system, we must divert the tool's communications with the user and the tool's requests for local resources. The NSW component which solves this problem is called the Foreman.

Finally, we expect that the output of one tool will be used as input to another tool. Unfortunately, if the first tool is a MULTICS editor and the second an IBM 360 compiler, this operation involves character translation (ASCII to EBCDIC), file reformatting (sequential file to blocked, recorded file), and file movement (across the Arpanet). To handle such file transformations and movements there is an NSW component called the File Package.

It is worth noting at this point that all of the above components are distributed. Every host in NSW has an MSG server process. Every site to which a user is connected has a Front End. Every tool bearing host has a Foreman. Every host on which NSW files are stored has a File Package. It is also worth noting that implementation details of these components vary from host to host. A MULTICS Foreman will be vastly different from an IBM 360 Foreman. Functional specifications for these components are fixed throughout NSW, but implementation and optimization decisions are left free.

Before proceeding to the NSW monitor, let us summarize the technical problems and the resulting components which provide the unified tool kit methodology.

o Inter-host inter-process communication	MSG
o User interface	Front End
o Diversion of communication with local operating system	Foreman
o File transformation and movement	File Package

2.2.2 NSW Monitor and File System

The design of the NSW Monitor - called the Works Manager - was probably more affected than any other component by the goals of NSW. Functionally it is not different from any other conventional access-checking, resource-granting monitor. Structurally, however, it is significantly different.

The goals of providing both large scale and reliability on conventional hardware led to the approach of distributing the Works Manager and file system. If there are many instances of the NSW monitor on many different hosts, then failure of a host is not catastrophic. Unfortunately, distribution runs counter to the problem-required logical unity of the monitor and file system. If a user inserts a file into the file system using one tool and one instance of the file system, and then requests the same file using a different tool and a different instance of the file system, the two instances of the file system must share a common file catalogue for the system to behave properly. Similarly, all instances of the monitor must share an access rights data base for proper validation of user requests to run tools.

A major technical problem in designing the Works Manager was that of creating synchronized duplicate data bases. The process structure of the Works Manager was designed so that such synchronization could be accomplished. Further, that process structure can handle the intrinsic distribution of NSW. Communication between the Works Manager and other NSW components - Front End, Foreman, and File Package - is via MSG, a relatively slow link.

2.3 Phases of NSW Development

The design and implementation of the National Software Works has proceeded in four slightly overlapping phases

- o Structural design and feasibility demonstration
- o Detailed component design
- o Prototype implementation
- o Reliability and performance improvement

In the following subsections we describe these phases in more detail.

2.3.1 Structural Design and Feasibility Demonstration

The first phase of NSW development began in July, 1974 and concluded in November, 1975. During this period, the basic architecture of NSW (described in Section 2.2) was established. Further, relatively ad hoc implementations of major components were made. These components were integrated into a system which was demonstrated to ARPA and Air Force personnel at Gunter AFB in November, 1975. This demonstration exhibited (functionally) various system functions, the user of batch tools on the IBM 360 and Burroughs B4700, the use of interactive tools on TENEX, transparent file motion and translation, and a primitive set of project management functions.

This demonstration confirmed that the expected NSW facilities could be implemented and that transparent use of a distributed tool kit was feasible. The NSW System, however, was inefficient and fragile. Further, many of the ad hoc implementations had design weaknesses which limited their general application to a sufficiently broad range of hosts and capabilities. For these reasons, an effort was begun to produce adequate component designs.

2.3.2 Detailed Component Design

This second phase of NSW development was begun in June, 1975 with the initial MSG design document. Specifications were developed for Tool Bearing Host components - MSG, Foreman, and File Package. All of these specification documents were completed by March, 1976. (They have all been revised since then, but the original specifications are still substantially correct.)

During the same period, the external specification of the Works Manager was also made. Again, although this specification has subsequently been revised, it is still substantially correct. The remaining portions of the core of NSW - i.e., the batch tool facility: Works Manager Operator, Interactive Batch Specifier, and Interface Protocol - were designed during phase one, and those designs were retained until phase four (see below).

The remaining major NSW component, the Front End was the subject of several design efforts. Three incomplete specification documents were produced but none of these was wholly satisfactory. Nevertheless, sufficient design to allow implementation of a functionally correct Front End was accomplished. Completion of a general specification for the Front End is one of the tasks remaining to be accomplished.

2.3.3 Prototype Implementation

As specification documents were completed, various contractors began implementation of the NSW components on the initial set of hosts - TENEX, MULTICS, and IBM 360. These efforts commenced in January, 1976. Implementation on TENEX proceeded more quickly than the efforts on the other hosts - primarily because the NSW system designers were also the TENEX implementors. By October, 1976 prototype implementations which conformed to the published specifications had been made for all TENEX TBH components. In addition, all components of the core system were available on TENEX.

Implementation of TBH components on MULTICS and IBM 360 proceeded more slowly; however, initial implementations of MSG components on both of these hosts were completed by the end of 1976. By November, 1976 sufficient progress had been made on implementation of a File Package and Foreman on MULTICS that it was possible to demonstrate an interactive tool running on MULTICS. Progress on implementation of 360 TBH components reached a similar position in September, 1977.

Also during this phase, a TENEX Front End which functionally supported the Works Manager and Foreman according to the appropriate specifications was implemented.

An NSW system containing prototype implementations according to the specifications of the core system, TENEX TBH components, TENEX Front End, batch IBM 360 tools, as well as a rudimentary MULTICS interactive tool was demonstrated to Air Force and ARPA personnel in November, 1976. At the same time, a demonstration of MSG components on all three hosts was also given.

2.3.4 Reliability and Performance Improvement

Even though implementation of components on MULTICS and IBM 360 was lagging, implementation of the core system, TENEX TBH components, and TENEX Front End had proceeded to the point that the issues of reliability and performance assumed major importance. The system exhibited sufficient functional capability that it could clearly support use by programmers if it were sufficiently robust and responsive.

The first task attacked was to provide robustness. Work had begun on a full-scale NSW reliability plan in 1975. The detailed plan was released in January, 1977. Since it was clear that implementation of the full plan was a major undertaking, a less ambitious interim reliability plan which ensured against loss of a user's files was begun in mid-1976. This plan was also released in January, 1977. By June, 1977 the core system, TENEX Foreman, and TENEX Front End had been modified to incorporate the features of that interim plan. In addition, both the MULTICS and IBM 360 Foreman

(only partially implemented) were altered to conform externally to the scenarios specified by the interim reliability plan. A system exhibiting the new scenarios was released for use in June, 1977.

Performance of NSW had been slow from the initial implementation. The reasons for slow response were many:

- o interaction between components was by a thin wire (MSG and the Arpanet).
- o NSW components (which constitute an operating system) nevertheless were executed as user processes under the local host operating system.
- o Component implementation had been oriented towards ease of debugging and other concerns of prototype systems rather than towards the performance expected of a production system.

In 1977, efforts to improve NSW performance were begun.

The first effort was the development of a performance measuring package for TENEX MSG. Results of the first set of measurements were reported in April, 1977. Some performance improvements were suggested by the initial measurements, but the most obvious suggestion was that more sophisticated measuring packages were needed. Several such packages were begun to perform various kinds of measurements on TENEX components. All of these packages were complete by February, 1978. By May, 1978, all TENEX components had been instrumented and measurements of page use, CPU time, elapsed time, use of JSYS (TENEX system commands), etc. had been taken under a variety of system load conditions and on several different TENEX hosts. Efforts are currently under way to implement the performance improvements suggested by these measurements. Performance improvements have already been made to several components. Results of these improvements are described in section 3.2 below.

Concurrent with the effort to improve NSW reliability and performance, an effort to make NSW a more packaged product were begun. Regression tests for the externally available NSW user system were developed and applied to each system release. A user's manual for the system was published. Documentation of the core system was produced. Finally, a draft configuration management plan was developed.

Phase four of NSW development is still continuing. Efforts to improve performance of TENEX components are substantially complete. Certain features of the full scale reliability plan have also been implemented, and phase four should be complete by mid 1979. Phase five, development of a production NSW system, is underway. The efforts proposed for phase five are described in section 4 below.

3. Current Status

3.1 Overview

The NSW system currently available to users was released in November, 1978. It has the following characteristics:

- Twenty interactive TENEX tools; five of these tools are installed in TOPS20, but some problems remain as compared to the TENEX installations.
- Ten interactive MULTICS tools, some of which are still being tested.
- One interactive IBM 360 tool, and nine IBM 360 batch tools.
- Basic set of system commands.
- User documentation and support.
- Rudimentary set of management procedures.
- Improved operability.
- Configuration of system includes following hosts:

  ISIC
  SRI-KA
  CCN
  RADC-TOPS20
  RADC-MULTICS

Functionally, the current NSW system is minimally adequate. It has a reasonable collection of tools, but many of these tools have not been adequately tested. The minimal set of user commands is available and tested, but many needed user features are lacking - e.g. command macros, '#' in file commands, I/O devices, Arpanet mail, etc. Performance has been improved significantly. The documentation of system components has been improved, but much needs to be done.

TENEX and TOPS20 are available as Works Manager or Tool Bearing Hosts according to specification, but TOPS20 tool encapsulation is currently less satisfactory than TENEX. Additional encapsulated tools can be installed in either environment to increase NSW capacity. Batch tools are available on the CCN IBM 360/91, and more can be installed as needed. A major overhaul of the entire batch system has made it more consistent with the rest of NSW, more flexible, powerful, operable and resilient. The IBM 360 Foreman implements only one interactive tool, and a minimal set of specified features. The MULTICS implementation has been improved enough to be included in the user system with an expanded tool set, although problems persist - particularly in the Foreman implementation.

The current status of the individual component implementations is presented in section 3, and planned improvements to the system are presented in section 4.

3.2 Components

In the following subsections we give a description of the current status of each NSW component.

3.2.1 Core System Components

The core system components - Works Manager, Checkpointer, and Works Manager Operator - are substantially complete. The Works Manager has been the object of an extensive and successful effort to improve its performance. The Checkpointer has had its functionality enhanced, and been made more robust. The Works Manager Operator has been substantially rewritten to interface to the Batch Job Package, and to conform to the coding standards imposed on the Works Manager.

3.2.1.1 Works Manager

At present, the Works Manager consists of a number of identical concurrent processes which implement the Works Manager procedures. All such processes share two common data bases, the Works Manager Table data base and the NSW File Catalogue. In addition to these processes there is a separate process, the Checkpointer, which makes periodic backup copies of the data bases.

The Works Manager supports 36 different Works Manager procedure calls, which are available to other NSW processes. These procedures are described in the Works Manager System/Subsystem Specification and the Works Manager Program Maintenance Manual.

A substantial effort was invested in implementing the scenarios described in the "Interim NSW Reliability Plan" (CA-7701-2111). These scenarios are as close as possible to the final NSW design which is described in "NSW Reliability Plan" (CA-7701-1411). The goal of these scenarios was to guarantee a user that a system malfunction -- other than catastrophic disk failure -- would cause few, if any, of her/her files to be lost. This guarantee includes files stored in the NSW file system as well as closed local files in a tool's workspace. It was not a goal to provide continuity of service in the face of individual component failure, nor was it a goal to eliminate long (possibly endless) waits by the user in the event of message delays or component failure (these desirable goals would be met by implementing the complete reliability plan).

In order to guarantee that NSW file system files not be lost (except under rare circumstances) it was necessary to preserve the NSW file catalogue. It was presumed that these files themselves are preserved by some mechanism on the file bearing host. Periodically (currently at approximate twenty minute intervals) the WM file catalogue is locked, the entire file catalogue is copied onto disk, and then the lock is released. The WM also maintains a data base of active users, active tools, etc., which is also copied onto disk (using the same mechanism described above for the catalogue). The Checkpointer, a new NSW component, was designed and implemented to fulfill this function.

The twenty minute interval introduced a window during which a file transaction may be lost if the WM host should crash. This twenty minute interval is sufficient with respect to NSW Exec commands. However, a tool might wait until termination to deliver any files; in this case, many hours of work could be lost. In order to avoid this problem, a mechanism was developed so that a Foreman could ensure the preservation of the local tool workspace (LND) in the event of either local host crash or the failure of other NSW components. The LND contains any files being delivered by the Foreman on behalf of the tool.

The mechanism developed ensured that the LND is preserved until after a file catalogue containing references to delivered files has been checkpointed. The LND is only (intentionally) erased after tool termination. Whenever a tool terminates normally, an additional message (FM-GUARANTEE) is sent by the Checkpointer (the process performing the file catalogue checkpoint) to every Foreman instance which terminated since the last checkpoint. Each Foreman instance sets a timer and if the FM-GUARANTEE message is not received when the timer goes off, the Foreman saves the LND.

The requirement for the Foreman is that it must be able to maintain the LND is such a way that it is preserved over Foreman host crashes. The Foreman must be able to explicitly invoke this save-the-LND mechanism. This allows the Foreman to explicitly preserve the tool's workspace should any difficulties arise during some scenario.

The AUTOLOGOUT scenario is initiated by a break in the connection between the user's terminal and the Front End. All running tools are forced to stop and initiate the save-the-LND mechanism described above.

A mechanism was also implemented which allows the user to have (some of) the saved files delivered to the NSW file system. This mechanism is provided by the LNDSAVED and RERUNTOOL sceanarios. Once a Foreman has performed the save-the-LND mechanism, it informs the Works Manager. The Works Manager maintains a record of such saved LNDs in each user's node record. A message will be sent to the user at each subsequent login until the user causes its deletion by using the RERUN command (which invokes the RERUNTOOL scenario). The user will receive messages about the saved LND until the user explicitly saves the files (TERMINATE subcommand) or deletes them (ABORT) subcommand). Currently, these are the only two options of RERUN which are implemented; it has been proposed that RERUN be expanded to allow the user to run a new instance of the tool in the saved LND.

A major change was the introduction of the Works Manager Table Facility as a performance enhancement. See Appendix A for details.

The Works Manager, which consists of approximately 25.7K lines of BCPL code, is structured into a number of layers. At the top level, WMMAIN waits for a procedure call message from another NSW process, does initial decoding and validity checking of any such message, then dispatches the message to the proper routine. The Works Manager Routines, WMRTNS, implement the 36 Works Manager Procedures. At their disposal are a number of lower-level utility packages and subsystems. The Works Manager Table Package, WMTPKG, handles all interactions with Works Manager tables. It serves as an interface to the Information Retrieval System, INFRTV, which manages the NSW File Catalogue and the Works Manager Tables. All NSW processes written in BCPL have available NSUPKG and BCPPKG. NSUPKG contains a number of facilities to handle MSG messages, create and record NSW fault descriptions, etc. BCPPKG provides basic utilities to handle character strings, do searching and sorting, and so forth.

As with other core system components and the TENEX/TOPS20 File Package, the Works Manager is transportable between TENEX and TOPS20 without modification. See Appendix C for details.

3.2.1.2 Checkpointer

The Checkpointer status mimics that of the Works Manager, since it consists largely of the entire Works Manager utility package, with a relatively small upper layer of code to implement the specific Checkpointer procedures. Thus, like other core system components, the Checkpointer is transportable between TENEX and TOPS20 without modification (see Appendix C). The performance improvements realized by the Works Manager table Facility also apply to some Checkpointer procedures.

The Checkpointer has the following characteristics:

- Implements the FM-GUARANTEE call on the Foreman required by the Interim Reliability Scenarios.
- Manages NSW file deletion. Files deleted by the user are actually deleted by the Checkpointer after a time interval, as required by the Interim Reliability Plan.
- Makes Checkpoint files of all Works Manager database files at approximately twenty minute intervals.
- Is robust and flexible to about the same level as the Works Manager itself.

3.2.1.3 Works Manager Operator

The modification/partial re-implementation of the Works Manager Operator (WMO) to meet the revised Batch Job Package specification included as Appendix B to this report is complete. The new version of WMO was relased as a component of the candidate user NSW system on November 16, 1978. The one WMO procedure specified is supported. this version has been extensively tested with the corresponding version of the CCN/360 BJP released on the same date, and there are no known outstanding deficiencies.

WMO shares a data base (the Job Queue File) with the Interacive Batch Specifier (IBS) module in the Works manager. We intend to remove this shared access by making all access to this data base be via procedure calls on WMO, which will have sole access. To this end, direct access to the data base by the WM to get a batch job status (NSw: JOB) has been replaced by a call on WMO by WM on the (new) WMO-SHOWJOB procedure. Direct access to the data base by IBS will be replaced by use of a WMO procedure, WMO-ENTERJOB, to be specified and implemented in the future.

The extensive modifications to WMO have allowed us to make its programming style consistent with that used in the Works Manager and File Package. Its use of the Works Manager utilities is also consistent with the other components, and its logging and timeout behaviour is identical.

Some able characteristics of the current WMO - particularly those not suggested by Appendix B - are as follows:

- o WMO is responsible for both processing the Job Queue File and handling WMO procedure calls. These two tasks are handled by distinct instances of WMO in any given NSW system.

  (1) There is exactly one instance of WMO processing the job queues. A standard locking discipline guarantees that precisely one such instance exists. This instance executes the job steps necessary to process a batch job, and initiates all procedure calls to external processes (WM, BJP, FP). It never receives generically addressed MSG messages.

  (2) There are zero or more instances of WMO which receive generically addressed MSG messages, and handle all currently defined WMO procedures. These instances never execute job steps or initiate external procedure calls. thus, these instance(s) provide external access to the data base.

- o A primitive retry mechanism exists. WMO will retry an external procedure call indefinitely when it fails due to network or remote host crash. It will retry a failed external procedure call a maximum of three times if the failure is due to resource problems, e.g. no disk space.

- o Status reports generated by WMO for display by WM (NSW: JOB) have been made more informative; all information supplied by BJP is reported.

- o The maximum number of jobs in the job queue file is currently 64. This may be increased when needed, but requires re-compilation and reloading of WMO.

- o The WMO cycle number may be set manually by the WMO utility (WMOUTL), but does not automatically increment with each cold start. "Cold Start" in this version occurs only when a new new job queue file is created.

3.2.2 TENEX/TOPS20 TBH Components

The TENEX/TOPS20 TBH is the most advanced of the three TBHs. All components (MSG, Foreman, and File Package) are substantially complete and tested. All components are transportable between TENEX and TOPS20.

3.2.2.1 MSG

The MSG specification was produced in January, 1976. It was revised in December, 1976 - primarily to resolve ambiguities in the earlier document. It was extended in April, 1978 to allow for support of multiple, concurrent NSW systems. The TENEX/TOPS-20 MSG component implements the revised and extended specification with only two exceptions (which are noted below).

The TENEX/TOPS-20 implementation of MSG is a single executable module which will run under TENEX, TOPS-20 Version 101B, and TOPS-20 Release 3. In addition to the communication functions supported for processes (and defined by the MSG-process interface specification) the TENEX/TOPS-20 implementation includes a powerful process monitoring and debugging facility, and comprehensive performance monitoring software.

The TENEX/TOPS-20 implementation does not perform MSG-MSG authentication. Message sequenceing and stream marking are not implemented (however the underlying software structure exists to support both).

The current implementation was extended to support new component initiation features required to support TOPS20 TBH components. In addition, a recent modification to MSG supports rapid timeout of attempts to contact remote hosts where an MSG is not up, or which are themselves down. This markedly reduces the wait time imposed on a user who has attempted to use an unavailable resource.

The implementation has also been modified to enhance its performance, based on extensive performance measurements completed this year. Changes include elimination of network connections for local message traffic, data re-structuring, reduction of calls on expensive JSYSES, and improved strategies for memory allocation.

3.2.2.2 Foreman

The current TENEX/TOPS-20 Foreman (Version 1521) implements all scenario functions defined by the interim NSW reliability plan in its most recent revision (March 1, 1977). The Foreman only supports tools which run in encapsulated mode. It does not yet support the direct use of NSW functions by any class of tools. It currently supports approximately twenty TENEX and five TOPS20 tools in this encapsulated mode. Some of these tools have been extensively tested and used within NSW; others have merely been superficially exercised.

The latest release can operate on both TENEX and TOPS-20 Release 3 configurations. There is a single .SAV file which detects at runtime the configuration type and modifies its behavior accordingly. This newest release has now had adequate field testing on the TOPS-20 machines. Not all TENEX NSW tools are available on TOPS-20 and those that are have not been tested to the same degree as their TENEX counter parts.

The current Foreman implementation handles the problem of storing "saved" tool workspaces through the temporary means of utilizing the workspaces themselves. A permanent facility to handle workspace management is already designed and implementation is pending.

The TENEX/TOPS20 Foreman has been extensively modified as a result of the extensive performance measurements made in early 1978 and reported in BBN report No. 3847, "A Performance Investigation of the National Software Works System". Performance enhancement has been currently limited to reducing resource consumption by the Foreman e.g. by minimizing use of expenive JSYSes, pre-allocating workspace directories, etc. Future work will address alternative system support configurations, and altered patterns of NSW communications.

3.2.2.3 File Package

The TENEX/TOPS20 File Package is now functionally complete. The task of writing Intermediate Language encode/decode for non-TENEX binary finished files is now complete, and has been tested with the CCN/360 File package for several representative binary file types. The current File package version has the following characteristics.

- All specified File Package procedures are implemented and tested for local, family, and non-family network transfers. Unspecified procedures to support the obsolete IP mechanisms in WMO have been expunged.

- The Intermediate Language (IL) encode/decode package has been re-structured for greater efficiency and maintainability. Encode/decode has been partitioned into three classes - text files, sequenced test files, and binary files; there is an encode and a decode module for each class, totalling six. Code size has increased, but both efficiency and code comprehensibility have been greatly enhanced. The interface between the (BCPL) calling routines and the (MACRO10/20) service routines has been simplified. Implementation of binary file encode/decode is complete, and has been extensively tested both against itself (i.e. against a remote TENEX simulating a non-TENEX host), and against the CCN/360 File Package. We have confirmed correct transmission of CMS2M object files from CCN/360 to TENEX/TOPS20.

- Performance enhancements have been implemented based on the results of BBN's performance investigation as reported in BBN report No. 3847, "A Performance Investigation of the National Software Works System", DRAFT VERSION, July 1978 by Richard E. Schantz. We have minimized the use of expensive JSYSes, notably the CNDIR (connect to directory) JSYS (average cost 220 ms per call). We have done so by specifying that the File Package must be able to create/read/delete files in its own filespace and Foreman workspaces without connecting to them, and letting it stay connected to its LOGIN directory. This has had no practical effect on the operation of NSW, beyond requiring that these directories be accessible from the system LOGIN directory. These enhancements hae resulted in a CPU usage reduction of up to 60% for delivery of a file from the Foreman workspace.

- The File Package is completely transportable between TENEX and TOPS20, requiring no modifications or patches. The simple transportability is based on the use of the Global Tailoring File for filespace name, logging information, and the use of the JSYS encapsulation packages now included in the Works Manager utilities. (See Appendix C).

- The logging of messages sent/received via MSG is under control of a switch in the Global tailoring File (as in WM, WMO and CHKPTR). When logging is disabled, CPU usage for typical FP calls is reduced 25% - 40%. For comparison, the FP retrieval calls analyzed in BBN report No. 3847, "A Performance Investigation of the National Software Works System", DRAFT VERSION, July 1978 by Richard E. Schantz,. which averaged about 2.9 ms, can be reduced to as low as 0.7 ms with logging disabled.

The File Package is written primarily in BCPL (approximately 6.9K statements including utilities.) The IL encode/decode package is written in Macro-10 and consists of approximately 1.7K instructions.

3.2.3 IBM 360 TBH Components

The IBM 360 TBH is the second most advanced host. MSG and the File Package are substantially complete. The Batch Job Package is debugged and available. The weakest component is the Foreman which implements only a small subset of the specification.

A new overlay mechanism which supports overlaying of exclusive segments has been constructed and installed in the File Package, Foreman, and Batch Job Package. This mechanism was required to allow these components to fit in available real core, and to allow for incremental increases in code size.

3.2.3.1 MSG

The IBM 360 MSG component implements substantially all of the revised MSG specification. It does not yet implement the April, 1978 extension. The features of the current version are:

- Flow control is implemented for both sides.
- The present TENEX limitation of 2048 bytes per message is larger than CCN can handle reliably with its current allocation of resources to the NCP region. Therefore, CCN's MSG is being configured with a maximum inter-MSG message size of 1024 bytes.
- An MSG process can be materialized automatically in either TSO or batch. The IBM 360 MSG requires that a process specifically "materialize" itself with a system call to the central MSG. Included in this materialization call is an event signal which will be signalled to perform the "termination signal" function; however, at present MSG-central never signals this event. No mechanism exists to allow a process which is restarting after it crashed (while MSG-central stayed up) to resume its earlier instance number.
- Both Sequencing and Stream Marking have been implemented.
- MSG now includes the ability to automatically start a process under TSO when MSG initializes itself after a system crash.
- Authentication is implemented in a manner which does not match the current specifications. The most important difference is that an ICP is required to the CCN authentication socket.
- Binary direct connections may use any byte size, but byte sizes smaller than 8 bits are likely to lead to problems in determining the actual length of the message.

- It has been decided to provide for a manifold of coexisting NSW systems on the same ARPANET hosts. This requires that a host support multiple MSG's, using different contact sockets. The 360 MSG was implemented to allow both a "production" and a "test" MSG to coexists, using different contact sockets.
  It is planned to modify MSG to allow more than two different MSG's to coexist; this modification is not as trivial as it was once believed to be.

- The current process interface for direct connections blocks internally, so that the process does not receive control from an alarm until all direct connection I/O completes. The direct-connection interface must be changed to be non-blocking.

- Now optimizes the number of idle server processes maintained based on predicted system load.

3.2.3.2 Foreman

The IBM 360 Foreman provides only a subset of the features defined in the specification, as only features required to support the DISPLAY tool are implemented. Specifically:

- The 360 Foreman supports encapsulated tools only; in particular, there is no Foreman-tool interface.

- Encapsulation does not extend to the file system. Therefore, NSW files can be fetched only before the tool starts. Files cannot be delivered, as this feature is not required by DISPLAY. This is accomplished by the Foreman interpreting a control stream which it receives in the "filename-list" field of the FM-BEGINTOOL command. A tool cannot dynamically select an NSW file.

- The only tool-control command implemented is FM-BEGINTOOL; FM-STARTTOOL and FM-STOPTOOL are not implemented. Any non-zero value for Entvec is interpreted as 1, i.e, it starts the tool at the beginning.

- There is no Local Name Dictionary (LND), and hence no saving of LND's. FM-OK is not implemented. No LND cleanup process is started automatically after a system crash. FM-REBEGINTOOL is currently implemented as another name for FM-BEGINTOOL. Otherwise, tool starting and stopping follow the interim reliability scenarios.

3.2.3.3 File Package

The IBM 360 File Package implements substantially all of the revised specification. A few features have either not been implemented or have been incorrectly implemented. Specifically:

- o All format effectors and record control tokens of IL are implemented. However, the variable format effectors HT, VT, LF, and FF, whose interpretations are defined for each file by the GFD are not fully tested with the Tenex File Package.

- o The IBM 360 File Package never arms itself for alarms, and it never sends an alarm. If an error condition is found during data transfer, the IBM 360 File Package will immediately close the connection (rather than send an alarm, as called for in the specifications). The File Package has no mechanism for reporting the status of a transfer operation.

- o The full Error Descriptors are not supplied by the File Package, due to PL/PCP restrictions. In particular:

  - The list of debug reports is always empty.
  - Only one error can be reported, the first one detected.
  - The values of the fault class and fault number fields have not been properly correlated with other File Package implementations.
  - The implementation of the Smithsonian Astronomical Date Standard is untested.

- o A format for family copies of files which cannot be described in IL has not been defined or implemented for the IBM 360 family. Hence, all net transmissions, regardless of family, use IL.

- o A local data set can be accessed by the File Package only if it exists within a directory in the NSW directory-group (i.e., having the NSW charge number). Since there is no mechanism to "connect" to a non-NSW directory, the password parameter is ignored.

- c [illegible] reblocking is not supported; a request to send an IL-encoded file with a transmission block size smaller than the IL blocksize in which it is recorded on disk may fail. This is not expected to be a problem, since File Package transmission block/sizes are expected to be established by gentleman's agreement and not varied.

- o Binary I.L. encode/decode has now been tested and debugged with the TENEX/TOPS20 File Package.

- o Only byte size 8 is supported for data transfer.

3.2.3.4 Batch Job Package

The initial implementation of the CCN/360 Batch Job Package is complete, and was released as a component of the candidate user NSW system on November 16, 1978. This implementation completely supports all BJP procedures specified in the revised Batch Job Package specification included as Appendix B to this report. This implementation has been extensively tested with the corresponding WMO version released on the same date, and has no known outstanding deficiencies. There are currently seven batch tools installed in NSW which may be run by WMO-BJP. Only the FORTRAN tool has been extensively tested and is known to run and produce good output. This testing deficiency is largely due to the circumstance that the personnel responsible for testing WMO-BJP are too unfamiliar with the other tools to create test input for them.

3.2.4 MULTICS TBH Components

The MULTICS TBH remains the weakest part of NSW. The components were implemented to comply only superficially with the specifications. The TBH components have been analyzed to a procedure Level, and a preliminary conformance study has been written. Enough problems have been fixed to justify the re-inclusion of MULTICS in the user system, with an expanded tool kit.

3.2.4.1 MSG

MSG is a relatively stable MULTICS component. Its biggest problem is its dependence on the unsupported TASKING software. Unsupported items in the specification, as documented on October 3, 1978 do not appear to compromise the usability of the MULTICS TBH software - many remain unimplemented in other TBH systems.

Configuration control has been improved by creating a contact socket table so that MULTICS MSG can contact remote NSW MSG's at the correct socket numbers.

3.2.4.2 Foreman

The Foreman contains the greatest number of unimplemented items, and is the source of most problems on the MULTICS TBH. The implementation suffers from the fact that it was implemented to support tools written specifically for NSW - i.e. tools that use NSW tool primitives - and only later extended to support tool encapsulation. In general, encapsulation can now be done, but the quality fo the encapsulation of each individual tool depends directly on the amount of work put into each encapsulation.

Specific improvements in the current implementation are:

- Many small bugs eliminated.
- Tool termination works essentially as specified.
- Alarm processing has been improved.
- More tools are encapsulated more reliably.

3.2.4.3 File Package

The File Package, like MSG, is a fairly reliable component. It conforms fairly closely to the specification, and supports file encodement into Intermediate Language about as well as the other TBH File Packages. Binary file transfer to non-MULTICS hosts is not supported, but is not required by any currently installed tools.

3.2.5 Front End

The COMPASS NSW Front End is not much different functionally then it was a year ago, since no major rewritings or addition of functions has been undertaken. It is, however, both faster and sturdier than it used to be:

Faster -- The FE program now handles (most of) its idle time by interrupt mechanisms rather than timed waits; hence it no longer consumes any CPU time between operations, and the CPU-time cost of waiting periods during operations has been cut in half.

Sturdier -- Anomalous conditions, especially in communications protocols, are detected more reliably and more discriminating responses are made. All known bugs have been corrected.

Several subtle accomodations have had to be made to the TOPS-20 operating system; but these have turned out to have no effect in the TENEX operating system, so that identical object-code files run on the two systems. Maintaining compatibility in this way means, of course, that no advantage has yet been taken of several of the advanced features offered by the newer TOPS-20.

Documentation:

The "external specs" of the FE are in reasonably good shape:

- The FE MSG Interface document, originally issued in November 1977, has been corrected and updated, and reissued in August 1978. It describes the format and content of all MSG messages sent or received by the FE.

- The "user interface" document -- the NSW User's Reference manual -- has been extended and partially rewritten, and was issued in November 1978 to describe the commands and operations available to the user in the NSW Version 3.1 release.

Shortcomings:

It is still possible for the FE process to "hang" if its conversational partner -- Works Manager or Foreman -- accepts an MSG message but then fails to reply. Without a moderately extensive rewriting of the programs, we are faced with the following choice in this circumstance:

(1) Abort the FE process, which leaves the user's Node Records in a blocked state so that he cannot log in again;

(2) Stop waiting for the reply and return to NSW command level: this appears to work for non-responsive Foremen, although the timeout has been set at 30 minutes; for Works Manager operations, this alternative leads to an out-of-synch situation from which the user cannot recover, if the belated reply does eventually arrive.

(3) Wait indefinitely for the reply, which is what we do now.

The program can still be made smaller and more efficient, and the input-editing facilities need to be completed.

3.3 NSW Performance

During the reported period a number of steps were taken to improve the overall performance of NSW. Three major avenues of approach were taken:

1. Memory use was monitored.

2. TENEX was monitored while running NSW in order to collect statistics on the gross use by NSW components of TENEX resources such as CPU time, JSYS monitor calls, and pager faults.

3. Detailed statistics were gathered on Works Manager CPU usage.

Memory use was monitored in two different ways. First, a memory monitoring tool called PAM was developed, and included in many NSW components. This tool, when activated, generates a map of exactly which virtual memory pages were accessed at least once between any two designated points in the execution of a program. This gives an accurate picture of the total number of memory pages that would be required to perform some NSW operation with no page faults. Because the result of using PAM is a map of exactly which pages were accessed, it is also possible to subdivide memory use into code and data accesses. From this it is possible to predict what the memory requirements would be for an NSW with a larger number of concurrent processes all of which shared code pages but each of which had its own local memory area.

PAM was able to show which pages were accessed at least once during an operation, but was unable to show how many times each page was accessed. Thus the figures obtained are doubtless larger than the true Working Set for NSW, in that pages are counted which may have been accessed only once or twice in an entire operation. In order to get a lower bound on NSW Working Set size, NSW was run on a metered version of TENEX and figures were obtained on the Working Set size that TENEX alloted to each NSW process. These figures represent a lower bound on the true Working Set, in that the figures also showed clearly that the TENEX configuration on which the tests were made had insufficient memory to run NSW without excessive paging. Unfortunately it is difficult to extrapolate from these figures just what the Working Set would be on a TENEX with adequate memory.

During the reported period BBN made a number of tests of overall system resource use by NSW. The results of these tests are described in great detail in

BBN Report No. 3847
A performance Investigation of the
National Software Works System
DRAFT VERSION
July 1978
Richard E. Schantz

In addition to the Working Set estimates already discussed, these tests showed that certain NSW processes were expending a great deal of time making JSYS calls to the operating system. As a result several NSW components, the File Package in particular, were altered to interact with the monitor more efficiently. This resulted in a substantial increase in File Package performance. These improvements are discussed in more detail in section 3.2.2.3 of this document.

These measurements of overall NSW component performance clearly showed that the Works Manager was consuming a large amount of CPU time, but gave no clue as to exactly where the time was being spent. To get a better picture of the problem a new performance tool for BCPL programs was developed: PFSTAT. PFSTAT takes samples of wall clock time, CPU time, and pager time at selected subroutine call and return points. The result is a detailed picture of what major subroutines were called and how much time each took to run. When PFSTAT was applied to the Works Manager it showed quite clearly that the major problem was that the Works Manager was using the powerful but slow Information Retrieval System to store all of its tables, including those tables which were accessed on every call. Accordingly, a new database management system called the Works Manager Table Facility was developed to hold the most active Works Manager tables, leaving the Information Retrieval System to handle only the NSW File Catalogue for which it was originally designed. As a result, the CPU time required by the Works Manager was reduced by a factor of 4. The Works Manager Table Facility is described in Appendix A of this document.

4. Future Directions

4.1 Overview

As noted in section 2.3, we are now in phase four of NSW development. The areas of greatest concern are improving reliability and performance. Substantial results in the area of performance improvement should begin to be visible in the user NSW system by October, 1978. Major effort on phase four whould be over by mid 1979.

The next phase of NSW development should be creation of a production NSW system. This system should exhibit the basic functionality already developed, as well as the robustness and responsiveness now being implemented. In addition, NSW needs to have the packaging, support, documentation, and capabilities of a finished production system. Phase five of NSW development will concentrate on providing these features. We expect to begin phase five in October, 1978 by beginning the expansion of the RADC TOPS-20 NSW to support the activities of NSW implementors. The first specific improvements scheduled are the installation and testing of tools needed by the implementors and the addition of an Arpanet mail facility. More details about specific features can be found in section 4.2.

In addition to program improvement, phase five will include the establishement of the administrative structure needed to support NSW users, manage the system configuration, operate systems, determine the priority of bug fixes and new features, prepare and distribute documentation, etc.

4.2 Components

In the following subsections we describe the tasks to be performed to complete phase four of NSW development and move into phase five.

4.2.1 Core System

4.2.1.1. Works Manager

Considerable effort must still be devoted to completion of phase four of Works Manager development. A number of measurements of Works Manager performance have been made and analyzed. Some improvements have already been made, and a substantial improvement is expected upon completion of the in-core Works Manager Table Facility (see section 3.2.1.1). More performance optimization is possible, and more effort should be devoted to measurement, analyses, and implementations. Current efforts at modeling should also be continued.

In addition, certain portions of the full scale NSW reliability plan should be implemented. While portions of that plan treated distributed data base synchronization, other parts dealt with issues of process and network failure and recovery. These other parts should be implemented. In particular, the try-retry mechanism and timing signals are needed. Moreover, a facility for archiving and restoring NSW files and data bases should be designed and implemented.

All of these performance and reliability improvements could be completed in 1979, thereby concluding phase four of NSW development. Phase five, which is concerned with "productizing" NSW should begin for the Works Manager in October, 1978. While the Works Manager is substantially complete, there are a number of extensions which should be made. These enhanced capabilities include:

- Arpanet mail interface - The procedures to support mail systems (e.g., Hermes) should be designed and implemented.

- Configuration management procedures - As noted in section 3.1, manual configuration management has already begun. As more NSW development work is done using NSW, it will be possible to automate configuration management.

- Direct file access - Use access and read access: Add two new kinds of NSW file access. Use access means that a user has undisputed rights to an NSW file. When he references the file he is given the NSW file copy - not a private copy. Any alterations he makes are immediately reflected in the file. Read access allows a user to read the actual NSW file copy - not a private copy. Thus it is suitable for data base files.

- Tool kits - When a user runs a kit of several tools on one host, the workspace should be left unchanged between each tool. Thus, intermediate files can be passed from tool to tool without delivery to NSW file space. Both of these features would greatly enhance and optimize the use of local tools.

- Version numbers - Design and implement a file version numbering facility. This facility must be rich enough to support configuration management within NSW.

- History file - Implement the Works Manager routines to record information on the History File. Design and implement at least some interesting management/accounting routines which access this file.

- Full file attributes - At present only the filename portion of the complete NSW filename can be used for retrieval. Also, the use of file attributes by tools is only permitted for the Global File Descriptor. The implementation of file attributes should be completed.

- Tool name extensions - The original concept of complete tool host transparency has proven unworkable. Thus, the notion of tool name should be extended to allow (explicit or implicit) host selection. By using the same mechanism as is used for files, the entire file lock system can also be used for tools.

- System status commands - The NSW user needs commands to interrogate system status and configuration: What tools are available? Which resources are up? What is the system load?

This list of WM extensions by no means exhausts the list of possible capabilities. These extensions could be scheduled for implementation in 1979; other features will undoubtedly be suggested as NSW implementors begin to use NSW for their own development efforts.

4.2.1.2 Works Manager Operator

Very little needs to be done to complete phase four of Works Manager Operator development. The mechanism used for batch job submission has proven to be reliable in the face of Works Manager, network, and batch host failure. Various detail improvements are required, but these will not consume much effort. Moreover, performance of the Works Manager Operator has not been a problem, since it operates in background mode. The elapsed time for its operation is only a miniscule fraction of total batch job execution time. Some effort should be devoted to carefully measuring and reducing CPU utilization because of the possible effect on interactive NSW components, but this is not a high priority task. Documentation of the Works Manager Operator should be completed in the near future.

In phase five, it will be necessary to extend the functional capabilities of the Works manager Operator. Such extensions include:

- Background file motion - The delays perceived by the user when files must be transferred or reformatted can be significantly reduced by performing such actions in background mode.

- Job chaining - A desirable extension is to allow multiple batch tools to be run in sequence. Such a sequence should not be limited to just one batch host.

- Device I/O - A variant of background file motion is to have WMO control input and output from devices local to a user.

- Support of small (or non-NSW) batch hosts - Some hosts may be too small to support a Batch Job Package. Also, some hosts may be desirable as batch hosts but may not have the required NSW components (MSG, File Package). The Works Manager Operator should be extended to use existing Arpanet protocols (FTP, RJE) to submit batch jobs to such hosts.

4.2.2 TENEX/TOPS-20 TBH

4.2.2.1 MSG

Very little additional effort is required for TENEX/TOPS-20 MSG. There are still some outstanding MSG design issues:

- Details of MSG-MSG authentication - The general mechanism is as specified in the MSG design document of December, 1976. However, the details of the ARPANET protocol exchanges are being re-examined.

- Maximum message size - The maximum message size is specified to be 65536 bytes (2**16). No implementation will accept messaged that large. At present there is informal agreement to limit message size to at most 2048 bytes.

- Process creation - This issue was skirted in the original specification. However, a satisfactory solution must be found which balances the dynamic cost of process initialization and the static cost of maintaining unused ready-to-run processes.

- Optimization techniques - Compound operations like "send then receive" should be added, and some MSG code could be included inside those processes run under MSG to reduce context switching.

- Reliability techniques - Allow for multiple hosts to be considered as recipients of generically addressed messages, so that the system can function better in the presence of "downed" hosts. The NSW Fault Logger is an example of a process which could make good use of such a feature.

Once these design issues are resolved, TENEX/TOPS-20 MSG must be modified to incorporate them. In addition, recent performance measurements have suggested a number of improvements which should be implemented.

4.2.2.2 Foreman

Completion of phase four for the TENEX/TOPS-20 Foreman involves two tasks. The first is the integration of the reliability mechansims described in the full scale NSW reliability plan - in particular, the try-retry mechanism and timing signals. The second task is improving Foreman performance with respect to CPU utilization and paging requirements. A number of such improvements have been suggested by the measurements and analysis already done. In addition, documentation of the Foreman must be produced. This documentation should be complete by February, 1979.

Although the TENEX/TOPS-20 Foreman substantially implements the specification, there are a number of additional capabilities which should be added. Some of these capabilities are implied by the specification, and some are additional. These capabilities include:

- Permanent integration of the TOPS-20 mountable structures interface
- Implementation of the solution to the saved LND workspace management problem
- Coordinated Works Manager/Foreman protocol design and implementation to have common data base items reflect local resource management decisions
- Implementation of tool-specific encapsulated tool interfaces to handle tool peculiarities and improve performance
- Direct tool interface to NSW functions - i.e., non-encapsulated tool interface
- Design and implementation of a Foreman modified for on-line tool debugging
- Design and implementation of Foreman extensions for tool kits.
- Incorporation of some of the file package's functionality in order to optimize file fetching and delivery operations.

File Package

Functionally, the TENEX/TOPS-20 File Package is essentially
e, including implementation of IL encode/decode for binary
Complete performance measurement and analysis must be done.
inary measurements have suggested some changes which should halve
ilization. Additional optimization should be performed. Some
concepts of the reliability plan could also be extended to the
ackage. The other major task to be completed in phase four is
tion of File Package documentation.

In phase five, the capability which should be added is
ization of cross-net file transfers. The baud rate of such
fers should be improved and automatic restart and backup
dures in case of file transmission errors should be designed and
mented.

4.2.2.3 File Package

Functionally, the TENEX/TOPS-20 File Package is essentially complete, including implementation of IL encode/decode for binary files. Complete performance measurement and analysis must be done. Preliminary measurements have suggested some changes which should halve CPU utilization. Additional optimization should be performed. Some of the concepts of the reliability plan could also be extended to the File Package. The other major task to be completed in phase four is production of File Package documentation.

In phase five, the capability which should be added is optimization of cross-net file transfers. The baud rate of such transfers should be improved and automatic restart and backup procedures in case of file transmission errors should be designed and implemented.

4.2.3 IBM 360 TBH

All IBM 360 components need to be documented.

4.2.3.1 MSG

The IBM 360 MSG should have the deficiencies mentioned in section 3.2.3.1 repaired. In addition performance should be measured and improved. As the MSG design issues mentioned in section 4.2.2.1 are resolved, the IBM 360 MSG should be modified to reflect those resolutions.

4.2.3.2 Foreman

The IBM 360 Foreman implements only a small subset of the Foreman specification. To the extent that there is user interest in interactive tools on IBM 360 hosts, the Foreman should be extended to implement the entire specification.

4.2.3.3 File Package

The IBM 360 File Package is essentially compiete. A few minor tasks remain to be done (see section 3.2.3.3), and these should be completed. Performance measurement, analysis, and improvement should be done. Optimization of cross-net file transfers should be done in conjunction with the TENEX/TOPS-20 File Package.

4.2.3.4 Batch Job Package

No further effort on this component seems necessary.

4.2.4 MULTICS TBH

As noted in section 3.2.4, the components of the MULTICS TBH have been baselined. It is now apparent that considerable effort must be devoted to making the Foreman implement the specification. MSG and the File Package implementations are operating according to specification. All MULTICS components need to be documented.

4.2.5 Front End

Functionally, the TENEX/TOPS-20 Front End is essentially complete. It has also been completely instrumented. Measurements have been taken and analyzed. While some level of ad hoc performance improvement is possible, the current Front End, which started as only a debugging tool, must be completely restructured in order to obtain a satisfactory level of performance. The Front End is implemented as a multi-fork process. Almost all of these multiple forks can be collapsed into a single fork. This will decrease both CPU utilization and space requirements. Front End documentation should also be completed.

An additional path toward optimizing Front End performance is to split the Front End into the "switcher" and "parser" functions. A document describing the functionality of the split was produced in July, 1978. Since this split is orthogonal to the current fork structure, the reduction of the number of forks should be completed before considering the implementation of the split Front End.

Parts of the full scale NSW reliability plan also must be implemented in the TENEX/TOPS-20 Front End - in particular, the try/retry mechanism and timing signals. With the completion of these performance and reliability tasks, phase four of Front End development will be finished.

There are several Front End enhancements which should be accomplished as part of phase five of NSW development. These enhancements include:

- o Optimization of local tool use - Some advantage should be taken when the Front End and task are on the same host. The split Front End is an approach to this optimization.
- o Macro facility - An NSW macro facility should be designed and implemented. This would permit users to execute a number of system/tool commands with a single command. It should be able to execute either online or in background mode.
- o User profiler - Use of the user profile to tailor terminal handling should be designed and implemented.
- o Access to text files - Currently the Front End can't access NSW files - if the user wishes a file listed, an editor or display tool must be invoked. The Front End should be able to list the file itself, and additionally should be able to take commands from a file to implement the "Runfile capability discussed later (see 4.3.3).

4.3 Functional Testing

4.3.1. History

COMPASS has been responsible since mid 1977 for functional testing of NSW as outlined in "National Software Works Test Plan", May 9, 1977, published by RADC/ISCP. Since that date, COMPASS has run a manual functional test script on each version of the NSW system which was a candidate for release as a new user system.

The initial version of this script was restricted to the level of test specified in RADC/ISCP Test Plan - to determine if NSW components functioned as specified in a friendly environment. Testing was limited to ensuring that all components in the test configuration (including remote TBH's) responded correctly to correct user input, and little effort was made to test the system in the face of incorrect input or errors in the system configuration. NSW systems tested to only this level tended to behave erratically. Therefore the functional test script was soon extended with a number of ad hoc tests of NSW's capacity to cope with user and configuration errors. This is the level of testing to which the candidate user system released on November 16, 1978 was subjected.

COMPASS has been mandated to develop and apply a more carefully designed and rigorous level of functional testing to future NSW system releases. The remainder of this section describes the direction for this future testing.

4.3.2. Functional tests - content

We define "functional testing" as follows: to determine whether a set of NSW components offered as a new system release meet the following requirements:

(1) Can be correctly configured as an operational NSW system, with all core and TBH components in a correct initial state for operation.

(2) All functions specified to be present in the release perform as expected for correct input, and all components in the configuration function as specified for correct input.

(3) All error detection and reporting functions work as expected for representative incorrect (user) input. All components report and recover from user induced errors as specified.

(4) The interim reliability scenarios perform as specified.

(5) The system recovers from configuration failures (e.g. TBH crashes) to the extent specified and expected for the release.

This testing includes complete tests for the delivery system for tools at each TBH - Foreman, File Package, Batch Job Package, etc - but does not cover acceptance of any tools.

The test scripts will be structured into a series of levels; the first level will test the least functionaity and the least complex core of the configuration. Each succeeding level will test more functionality and/or more of the system configuration.

The general contents of the scripts will be as follows:

Level 0: Set up the complete system configuration, and verify that all components are in a proper initial executive and communications state.

Level 1: Test core system: all components local on Works Works Manager host.

(a) Test all possible NSW command paths with correct input in the following order:

i. LOGIN, MOVE, CHANGE password, LOGOUT.

ii. Project management tools: nodes, assign rights, etc.

iii. ALTER comamnd - SCOPE manipulation.

iv. File commands - NET, RENAME, COPY, DELETE, SEMAPHORE. Local file transfers only.

v. Enter a batch job. (Processing deferred).

vi. Use a local interactive tool. Test slewing, multiple tools, RESUME.

All recognition and completion features of the Front End are to be tested.

(b) Recapitulate relevant sections of (a), with representative errors on input. The error detection and reporting facilities of the local components are to be tested in the following order:

i. Front End
ii. Works Manager
iii. File Package
iv. Foreman

(c) Where appropriate in (a) and (b), the operation of the Checkpointer is to be monitored, and message and error logging is to be monitored.

Level 2: Test the distributed system: at least one instance of each TBH family to be involved in the configuration.

(a) File transfer tests

i. Test family transfers, where available. Currently limited to TENEX/TOPS20 hosts due to lack of multiple host resources.

ii. Test non-family transfers. At least one text file transfer back and forth between each family pair in configuration, and one round-robin transfer in a chain including all families. Multiply translated files must be identical under Intermediate Language semantic specification. At least one binary file transfer of each defined type.

(b) TBH test

i. Execute a batch job at each BTBH. Monitor performance of Works Manager Operator and Batch Job processor for each job.

ii. Execute one interactive tool at each TBH. Level of test identical to 1 (a) vi.

(c) Recapitulate (a) and (b) introducing representative errors in user input.

Level 3: Test interim reliability scenarios. Induce each error condition covered by interim reliability plan, and monitor all components involved for correct behavior.

(a) Initial test will be for the core system only, particularly to test correct behavior of Works Manager.

(b) Test of Foreman capability for each TBH. Induce only those failures which test the Foreman's role in the reliability scenarios.

Level 4: Test system response to induced configuration failures. Beyond checking response to "crashed" TBH (NSW taken down), the content of this test level is to be specified.

4.3.3. Functional tests - methodology

It will be necessary to automate these tests as much as possible both to avoid expending excessive professional staff time on them, and to make the tests reliably repeatable. COMPASS has investigated three classes of tools which can assist this automation effort:

1. Run file facilities external to NSW:

   TENEX RUNFIL
   TOPS20 TAKE
   TELNET take.input.from.file

2. Run file facilities within NSW.

   Front End RUNFILE command

3. Production (syntactic rule) systems

   RITA

1. Run file facilities external to NSW

The tools listed are all basically similar. Each has the advantage of being familiar, tested and straightforward. All lack a sufficiently sophisticated means of synchronizing their input to the processes they control with what is in fact happening. The synchrony problem limits these tools to situations in which no slewing between TELNET connections is done. This excludes any testing of NSW tools, and makes changing TELNET conversational partners to monitor configuration status changing unreliable.

2. Run file facilities within NSW

Provision of a RUNFILE command has one outstanding advantage: the Front End is always aware of the identity of the user's conversational partner-NSW command processor, HELP call, or tool - and is thus perfectly placed to control the synchronization of command file with the actual behavior of NSW. An additional advantage is that we can add desired features to this facility as needed, but must accept the others as they are. The disadvantages are that this facility has to be designed, implemented and tested; and that it can only automate the user input portions of the test scripts.

3. Production systems - RITA

RITA has the advantage that it can handle both user input and configuration management with a sufficiently rich rule set. Our studies indicate that the development of such a rule set would be a demanding job. A more significant problem is that TENEX RITA is likely to consume excessive CPU recources to run a rule set as complex as that needed by NSW.

Proposed Methodology

We propose that a mixture of manual testing and the use of two of the tools described above be used to run the functional tests. The mix would be as follows:

1. Use RITA to set up and initialize the NSW configuration for each level test, and confirm that the initialization is correct.

2. Use NSW RUNFILE to automate all user input to test Levels 1, 2, and 3. The RUNFILE facility will have some or all of the following features:

   (i). Ability to interrupt

   (ii). A synchronization scheme

   (iii). HELP from attached user if synchronization failure occurs

   (iv). A PAUSE feature

   (v). A macro feature - string and/or file name binding at run time.

   (vi). A "learning" feature which will allow the Front End to do most of the work of turning a manual script into a command file (speculative).

3. Use manual scripts for much of level 3 testing and most of level 4 testing. Probe system status and monitor component operation as required during Level 1, 2, and 3 testing.

4.4 Miscellaneous

There are additional tasks to be undertaken which do not fall within the scope of a single component. One major effort, the creation of an administrative structure for NSW, was mentioned in section 4.1. In this section we list some additional efforts:

- Help facility - an online help mechanism for NSW users should be designed and implemented. This should probably look like a tool within NSW.

- Distributed system debugger - It should be possible to debug a distributed system like NSW from within NSW. An appropriate debugger should be designed and implemented. This will almost certainly require changes to the Works Manager and Foreman components, and possibly to MSG also.

- Fault logger - An NSW wide component for logging all error messages should be designed and implemented.

- Automated testing - The functional and stress/regression testing of NSW test and user systems should be automated.

- Management tools - Tools for manipulating the project tree are available in rudimentary form. These should be improved, and additional tools for accessing the History file, report generation, etc. designed and implemented.

- Operators tools - A tool kit for the user system operator to at least partially automate data base cleanup, system starting, etc. should be designed and implemented.

- Tool installation - Install, test, and document more NSW tools. In particular, install a tool kit adequate for NSW implementors.

Bibliography

NSW Protocol Committee. MSG: The Interprocess Communications Facility for the National Software Works (Preliminary). Massachusetts Computer Associates, Inc. CADD-7601-2611, and Bolt, Beranek and Newman, Inc. 3237, January 23, 1976.

Muntz, C.A., and Cashman, P.M. File Package: The File Handling Facility for the National Software Works (Preliminary). Massachusetts Computer Associates, Inc. CADD-7602-2011, February 20, 1976.

Schantz, R.E., and Millstein, R.E. The Foreman: Providing the Program Execution Environment for the National Software Works (Preliminary). Bolt, Beranek and Newman, Inc. 3265, and Massachusetts Computer Associates, Inc. CADD-7604-0111, March 31, 1976.

Geller, D.P. NSW User's Guide (Preliminary). Massachusetts Computer Associates, Inc. CADD-7605-3111, May 31, 1976.

Geller, D.P. NSW Manager's Guide (Preliminary). Massachusetts Computer Associates, Inc. CADD-7605-3112, May 31, 1976.

Warshall, S. NSW: Tools for Management Support (a final report). Massachusetts Computer Associates, Inc. CADD-7607-0111, July 1, 1976.

NSW Protocol Committee. MSG: The Interprocess Communication Facility for the National Software Works (1st revision). Massachusetts Computer Associates, Inc. CADD-7612-2411, and Bolt, Beranek and Newman, Inc. 3237, December 24, 1976.

Cashman, P.M., and Faneuf, R.A., and Muntz, C.A. File Package: The File Handling Facility for the National Software Works (1st revision). Massachusetts Computer Associates, Inc. CADD-7612-2711, December 27, 1976.

Schantz, R.E., and Millstein, R.E. The Foreman: Providing the Program Execution Environment for the National Software Works (1st revision). Bolt, Beranek and Newman, Inc. 3442, and Massachusetts Computer Associates, Inc. CADD-7701-0111, January 1, 1977.

Shapiro, R.M., and Millstein, R.E. NSW Reliability Plan. Massachusetts Computer Associates, Inc. CA-7701-1411, January 14, 1977.

Millstein, R.E., and Shapiro, R.M. Interim NSW Reliability Plan. Massachusetts Computer Associates, Inc. CA-7701-2111, January 21, 1977.

Bearisto, D.B. The Front End: User's Interface to the National Software Works. Massachusetts Computer Associates, Inc. CA-7701-2112, January 21, 1977.

Millstein, R.E., and Shapiro, R.M. Interim NSW Reliability Plan (1st revision). Massachusetts Computer Associates, Inc. CA-7701-2111, February 8, 1977.

Millstein, R.E., and Schantz, R.E. NSW Tool Builder's Guide (1st revision). Massachusetts Computer Associates, Inc. CADD-7702-1811, and Bolt, Beranek and Newman, Inc. 3308, February 18, 1977.

Millstein, R.E., and Shapiro, R.M. Interim NSW Reliability Plan (2nd revision). Massachusetts Computer Associates, Inc. CA-7701-2111, March 1, 1977.

Shapiro, R.M., and Millstein, R.E. NSW Reliability Plan (1st revision). Massachusetts Computer Associates, Inc. CA-7701-1411, June 10, 1977.

Guerrieri, M., Schaffner, S., and Sluizer, S. Works Manager Program Maintenance Manual. Massachusetts Computer Associates, Inc. CADD-7709-2711, September 27, 1977.

Guerrieri, M., Schaffner, S., and Sluizer, S. Works Manager Subsystem Specification. Massachusetts Computer Associates, Inc. CADD-7709-2712, September 27, 1977.

Schaffner, S. Works Manager Database Specification. Massachusetts Computer Associates, Inc. CADD-7709-2713, September 27, 1977.

Bearisto, D. System/Subsystem Specification; WMO & IBS Methodology; WMO & IBS Documentation Program Maintenance Manual; WMO & IBS System Documentation. Massachusetts Computer Associates, Inc. CADD-7709-2714, September 27, 1977.

Geller, D.P., and Sattley, K. NSW User's Reference Manual System Version 2.1. Massachusetts Computer Associates, Inc. CADD-7710-2611, October 26, 1977.

Schaffner, S., Sluizer, S., and Guerrieri, M. NSW Utilities Program Maintenance Manual. Massachusetts Computer Associates, Inc. CADD-7804-0111, April 1, 1978.

Sluizer, S., Guerrieri, M., and Schaffner, S. Information Retrieval System Program Maintenance Manual. Massachusetts Computer Associates, Inc. CADD-7804-0112, April 1, 1987.

Glossary

NSW	National Software Works
WM	Works Manager
WMO	Works Manager Operator
TBH	Tool Bearing Host
MSG	NSW Interprocess Message System
FM	Foreman
FP	File Package
FE	Front End

Works Manager Table Facility

As of November 1977, NSW had progressed to the point where it was sufficiently robust and complete to allow serious use. However, it was very slow, even with only one or two users logged in. More than two users was clearly out of the question.

It was felt that part of the problem was due to the difficulty of implementing an interprocess protocol such as MSG on top of the standard TENEX monitor. In addition, it was known that the physical memory on the host machine was inadequate to support a minimal NSW working set. It was also clear, however, that a great deal of the problem was simply that it took a lot of CPU processing to perform any NSW user command. In particular, the Works Manager required a lot of CPU time.

Some relatively simple changes were made in 1977 that speeded up the Works Manager by a factor of two. As a result, Works Manager CPU usage per procedure call was now comparable to that of other NSW components. Unlike the File Package, the Foreman, and WMO, however, the Works Manager participates in almost every interaction with the user. Thus while the Works Manager was not always the worst CPU time burner per call, it was certainly the worst per user session due to the large number of calls made on it.

To give some perspective on the problem, figure 1 shows the amount of CPU time consumed by the 1977 Works Manager during the indicated procedure calls.

A-2 CPU TIME REQUIRED TO PERFORM WORKS MANAGER PROCEDURE CALLS

MARCH 1978

Procedure	CPU time (seconds)
LOGIN	1.2
RUNTOOL	2.0
PUT	2.8

Fig. 1

By March of 1978 it became possible for us to consider making a concerted attack on the problem of Works Manager performance. Not only was there manpower available to work on the problem but also PFSTAT, our performance measuring tool for BCPL programs, had been developed to the point where it was adequate to the task of pinpointing the sources of the problem.

We made a number of tests of various Works Manager procedures, and a pattern quickly became apparent: the Works Manager was spending most of its time making calls on the Information Retrieval System. This confirmed what we had already suspected, since earlier tests on the Information Retrieval System had shown that table access calls were so expensive by themselves that there would be little CPU time left in most Works Manager routines to assign to any other cause.

The Information Retrieval System itself will need some substantial optimization sometime in the future. The primary problem in March of 1978, however, was simply that the Information Retrieval System was originally designed to support only the NSW File Catalogue, and the File Catalogue has substantially different characteristics from other Works Manager tables. Figure 2 contrasts the differences between the NSW File Catalogue and the other Works Manager Tables.

CHARACTERISTICS OF NSW FILE CATALOGUE

1. Very large number of items to store.
2. Infrequent access to any one item.
3. Retrieval by keyword.

CHARACTERISTICS OF OTHER WORKS MANAGER TABLES

1. Small number of items -- most items could fit in process virtual memory for a medium sized (100 node) NSW.
2. Frequent access to many items.
3. Retrieval not only by keyword but also by parameter value.

The NSW File Catalogue contains a potentially large number of items. Generally, no individual item in this database will be accessed very often. Item retrieval is by keyword: "Give me all files whose names start with COMPASS.SLUIZER and which also contain WALDO". Because people are relatively inventive, there is expected to be a large number of keywords.

However, the other Works Manager tables fit quite a different pattern. They contain a relatively small number of items. In fact, for an NSW of moderate size, say 100 user nodes, all these tables could fit into part of a process virtual memory. Most of the items in these tables will be accessed frequently. Finally, the name structure of these tables is different. Most items have simple names of fixed structure. Furthermore, item name elements must at times be used as parameter values instead of as keywords. For example, the Works Manager may wish to retrieve all Deleted File Entries that have timestamps which are at least 20 minutes old.

It is felt that the best overall course of action was to design and build a facility expressly for Works Manager Tables -- the Works Manager Table facility. This package would be tailored specifically to the particular characteristics of Works Manager Tables. First, all retrieval information would be stored online, i.e. in process virtual memory. Second, as many table entry bodies as would fit would also be online. Third, retrieval would be done on a fixed name form and the retrieval would be based on parameter values instead of keywords.

We were forced, however, to take account of two implementation problems: First, there was at best one person available for at most 6 months to design, implement, and test any new facility. Second, there were even more stringent limits on the manpower available to change higher-level parts of the Works Manager in order to use the new facility. Thus the new interface could not be radically different from the old one.

Of course, if these restrictions had proven unworkable then the projected schedule could have been altered. Happily, it was possible to create a reasonable design that would still allow an implementation within the time limit.

The time schedule was met in the following manner: First, the design was deliberately made quite standard. The algorithms and data structures used were ones which were known to be simple, proven, and flexible. NSW overall may be a research project, but the Works Manager Table Facility certainly was not. Second, once the design was complete the implementation was done to cost. Where time did not permit an optimal implementation of some part of the design, simpler data structures and algorithms were employed which could easily be replaced later. For example:

1. Exclusive locks were used for concurrency control. Now, an exclusive lock provides excessive protection; for example, it prevents two separate processes from reading the same element simultaneously. However, the way the database is structured there should be few collisions.

2. Singly-threaded lists and linear scans were used instead of more complex structures and faster scans. As the figures will show later, the Works Manager Table Facility that resulted was still adequately fast.

3. Fixed allocations were used. For example, whenever an online database is set up the maximum number of table entry slots needed must be preallocated. This wastes space in the database. Code could be added later which could dynamically reconfigure a table header whenever the need arose, allowing preallocated table header space to be made much smaller.

Finally, the time schedule was met by deferring implementation of parts of the design not needed immediately, in particular overflow storage and checkpointing. These parts were implemented later, in version 2.

In evaluating these implementation shortcuts, we must remember that the goal is a fast Works Manager, not necessarily a fast Table Facility. For example, we could create a version 3 with dynamic reconfiguration of table entries in order to save table space. However we could use the same time instead to make changes to other parts of the Works Manager. These other changes might well save a great deal more table space for about the same coding effort.

Version 1 of the Works Manager Table Facility was first used in a Works Manager in August of 1978. At the highest logical level, it implements a table structure similar to the one already supported by the Information Retrieval System. The major differences between the two systems are confined to lower logical levels.

The database consists of a number of Works Manager Tables. Each table consists of a set of table entries, for example Active User Entries or Node Entries. A table entry is the basic unit of transaction in the Works Manager Table Facility. An entry is composed of:

1. An entry name consisting of a list of parameter values.

2. A body, which at this level of detail is nothing more than a block of arbitrary data.

3. A set of external locks. These locks are used by Works Manager instances to coordinate their use of table entries. They are not used by the Works Manager Table Facility itself.

Parameters in an entry name can be character strings, integers, or timestamps. In the Information Retrieval System, all parameters were character strings. The shape of an entry name, that is, the number of parameters and the type of each, varies from table to table but is the same for all entries in a given table. In the Information Retrieval System, on the other hand, different entries in the same table could have different numbers of parameters. In both systems, however, the entry names are the sole means of choosing one entry over another in a retrieval.

The online database is a single block of data, kept in a single TENEX file. All processes that access this database map a portion of their virtual memory down onto this file. Thus each process sees the database as a part of its own memory.

This single block of memory is divided into variable-sized blocks. Each block is kept track of by a two word header which is separate from the block. There is a singly-threaded list of free blocks arranged in order of increasing address in memory. When the database is first created, this list contains one single, large block. As memory is used and then later released, the list will grow. To satisfy a request for memory, the list is scanned to find the smallest block that is large enough. Generally, the requisite memory is then split off from the block unless the block is only slightly larger to begin with. The address of the block is returned, not the address of the header that defines it.

Blocks which are in use are threaded onto another list. This list is necessary in order to find the block header when the block is later freed, as higher-level routines know only the address of the block, not the address of the header.

One of the more significant differences between the Works Manager Table Facility and the Information Retrieval System is that in the Works Manager Table Facility the information which defines a table is centralized, whereas in the Information Retrieval System this information is distributed. In the online database each table is represented by a data structure called a table header. The header consists of a fixed part which contains several items, including a definition of entry name shapes and the starting address and length of a block of entry slots. All slots are the same size. Some slots are marked as not in use; the rest define table entries. A slot in use contains the following information:

1. The value for each parameter in the entry name.

2. The entry external locks.

3. A pointer to the body.

The original design was based on the idea that a Works Manager process would be given the address of an entry body, providing Works Manager processes with the most direct access possible to table entries. However, implementers of Works Manager procedures felt that this was too dangerous. All access is now through copies, just as it was in the Information Retrieval System. This requires an extra block transfer operation for most table accesses. This consumes on the order of an extra millisecond of CPU time, and is not really a substantial contribution to system overhead.

Version 1 of the Works Manager Table Facility was first used in a Works Manager sometime in July of 1978. This initial version put the Active User Entries and the User Identification Entries online and left all other tables in the Information Retrieval System databases. These two tables are referenced in almost every Works Manager Procedure call. Figure 3 shows clearly that even this small change produced a substantial improvement. This more-or-less served as an acceptance test of the overall concept, and soon thereafter all other Works Manager tables except the NSW File Catalogue were put online. The final figures, given in the right hand column, were just about what we had hoped to see, given our PFSTAT runs of 1977. As you can see, routines which do not access the NSW File System take about one third of a second, while routines which do access the file system take about 1 second. We have used PFSTAT to analyze the difference between GET and PUT which, on the surface at least, should take almost exactly the same amount of time. We have found a minor problem in the interface between the Information Retrieval System and the Works Manager. When this problem is corrected, we expect that GET and PUT will both take about 800 milliseconds.

Version 1 of the Works Manager Table Facility was released for system testing late in 1978. Version 2, released for testing in October of 1978, implemented those features of the design which were deferred, namely checkpointing and overflow handling. Specifically, the new features in version 2 are:

1. Overflow handling -- when space is needed online, least-recently-used table bodies are dumped into an Information Retrieval System database. The item number of the offline body rather than the item name is stored in the entry slot, to avoid an expensive keyword search when the item is later accessed.

2. Checkpointing -- A checkpoint lock was added to ensure that while the Checkpointer is copying the database, no process can be writing into that database.

3. Dynamic allocation of memory block headers -- This reduces space wastage and removes an arbitrary restriction on the degree to which memory can be broken into separate blocks.

4. Various improvements to increase robustness and ease of use, for example, database internal version numbers and support for DBSTAT.

These performance improvements were sufficient to solve the immediate problem. Eventually, however, another optimization pass will be needed. It would be premature to say much about what optimizations should be performed next, as we haven't had a chance to think all that deeply about the problem yet. The first step would probably be to use PFSTAT to generate quantitative models of present system performance. From this we could predict the overall effect of any specific change before we actually made that change.

To give some feel for the current status of the Works Manager and to show how we might go about making another optimization pass, let us examine figure 4, which shows some PFSTAT output from a call on the Works Manager Procedure WM-LOGIN.

no.	type	lvl	caller PC		called PC		realtm	runtm	dlreal	delrun	dlpage
45	call	2	WMMAIN	11466	M3PRMS	325027	56478	905	0	0	0
46	retn	2	WMMAIN	11466	M3PRMS	325027	178312	918	121834	13	42
55	merg	2	WMMAIN	11652	WMLOGS	37527	179729	962	1417	44-18	56
56	call	3	WMLOGS	37577	D8IREP	316067	179730	963	1	1	0
57	retn	3	WMLOGS	37577	D8IREP	316067	179746	979	16	16	0
58	call	3	WMLOGS	37655	OPNUTS	251731	179954	983	208	4	6
59	retn	3	WMLOGS	37655	OPNUTS	251731	181176	1015	1222	32	80
60	call	3	WMLOGS	40236	PASHZN	342055	181207	1018	31	3	1
61	retn	3	WMLOGS	40236	PASHZN	342055	181209	1020	2	2	0
62	call	3	WMLOGS	40426	DRTUTS	274556	181349	1023	140	3	17
63	retn	3	WMLOGS	40426	DRTUTS	274556	181608	1029	259	6	6
64	call	3	WMLOGS	40457	DRTUTS	274556	181610	1031	2	2	0
65	retn	3	WMLOGS	40457	DRTUTS	274556	181616	1037	6	6	0
66	call	3	WMLOGS	40504	DRTUTS	274556	181618	1039	2	2	0
67	retn	3	WMLOGS	40504	DRTUTS	274556	181624	1045	6	6	0
68	call	3	WMLOGS	40531	DRTUTS	274556	181626	1047	2	2	0
69	retn	3	WMLOGS	40531	DRTUTS	274556	181630	1051	4	4	0
70	call	3	WMLOGS	40556	DRTUTS	274556	181632	1053	2	2	0
71	retn	3	WMLOGS	40556	DRTUTS	274556	181639	1060	7	7	0
72	call	3	WMLOGS	40603	DRTUTS	274556	181641	1062	2	2	0
73	retn	3	WMLOGS	40603	DRTUTS	274556	181647	1068	6	6	0
74	call	3	WMLOGS	40630	DRTUTS	274556	181649	1070	2	2	0
75	retn	3	WMLOGS	40630	DRTUTS	274556	181652	1073	3	3	0
76	call	3	WMLOGS	40655	DRTUTS	274556	181654	1075	2	2	0
77	retn	3	WMLOGS	40655	DRTUTS	274556	181751	1159	97	84	0
78	call	3	WMLOGS	40702	DRTUTS	274556	181753	1161	2	2	0
79	retn	3	WMLOGS	40702	DRTUTS	274556	181756	1164	3	3	0
80	call	3	WMLOGS	40726	DRTUTS	274057	181758	1166	2	2	0
81	retn	3	WMLOGS	40726	DRTUTS	274057	181760	1168	2	2	0
82	call	3	WMLOGS	40751	DTBSBS	177621	181871	1171	111	3	5
83	retn	3	WMLOGS	40751	DTBSBS	177621	182122	1176	251	5	10
84	call	3	WMLOGS	40766	WMEUTS	254640	182124	1178	2	2	0
85	retn	3	WMLOGS	40766	WMEUTS	254640	182332	1191	208	13	10
86	call	3	WMLOGS	41016	DRTUTS	276304	182351	1193	19	2	1
87	retn	3	WMLOGS	41016	DRTUTS	276304	182353	1195	2	2	0
88	call	3	WMLOGS	41045	DRTUTS	274057	182355	1197	2	2	0
89	retn	3	WMLOGS	41045	DRTUTS	274057	182357	1199	2	2	0
90	call	3	WMLOGS	41071	OPNUTS	253330	182405	1202	48	3	3
91	retn	3	WMLOGS	41071	OPNUTS	253330	182629	1340	224	138	8
92	call	3	WMLOGS	41110	WMUTIL	73251	182675	1344	46	4	3
93	retn	3	WMLOGS	41110	WMUTIL	73251	182761	1366	86	22	10
94	call	3	WMLOGS	41117	BCPMCH	342625	182763	1368	2	2	0
95	retn	3	WMLOGS	41117	BCPMCH	342625	182849	1373	86	5	3
96	call	3	WMLOGS	41153	WMMAIN	11677	182851	1375	2	2	0
97	retn	3	WMLOGS	41153	WMMAIN	11677	185436	1434	2585	59	35
98	call	3	WMLOGS	41167	DRTUTS	276304	185541	1439	105	5	14
99	retn	3	WMLOGS	41167	DRTUTS	276304	185543	1441	2	2	0
100	retn	2	WMMAIN	11652	WMLOGS	37527	185544	1442	1	1	0
113	merg	2	WMMAIN	11466	M3PRMS	325027	186156	1470	612	28-26	30

Fig. 3

A detailed description of how PFSTAT works is beyond the scope of this document, but briefly PFSTAT manipulates the BCPL control stack in order to take samples on selected subroutine calls and returns. It will sample all calls down to some specified level in the runtime call tree. Below this level PFSTAT is disabled, so there is no overhead on low-level calls. The global sampling level can be manipulated locally by specifying that certain calls are to be magnified or pruned. In this particular case, the global level is 2 and the call from the Works Manager main program, WMMAIN, to the login subroutine has been flagged for magnification. The listing shows real (or wall clock) time, runtime (or CPU time), and paging time, all in milliseconds. The full listing for this run is quite a bit larger. We have used PFSTAT's formatted dump facility to print only a portion of that listing. In addition, we have merged uninteresting sequences of nodes. This merger is done by the formatter, after the samples are taken.

For example, consider samples 76 and 77 of figure 3. Sample 76 was taken when the Works Manager called a subroutine. The call was from address 40655 which is in module WMLOGS. In fact, we know that it is in the WM-LOGIN routine. We could use the BCPL debugger, BDDT, to print out the line of BCPL code that made the call. The subroutine being called is at address 274556 in module DRTUTS. This subroutine turns out to be a utility routine which copies table elements. Again, we could examine the subroutine with BDDT, given the address. Sample 76 was taken 181.654 seconds of real time after PFSTAT was enabled and the sampling began. By this time 1.075 seconds of CPU time had been consumed. A real time of 181.654 seconds is 2 milliseconds of real time after sample 75 was taken. During this interval, 2 milliseconds of CPU time and 0 milliseconds of paging time were consumed. In other words, this process had complete use of the CPU during the interval and no pages had to be read in.

Sample 77 was taken when this subroutine returned. The real time was 181.751 seconds, or 97 milliseconds later than the real time at sample 76. By now, 1.159 seconds of CPU time had been charged to this process, which was 84 milliseconds more than had been charged at sample 76. No additional paging time was charged, indicating that the entire subroutine was already in real memory.

We have found PFSTAT to give highly repeatable results, which are not noticeably affected by system load, amount of real system memory, or type of scheduler. The only real problem is that the sampling procedure itself takes, on the average, just under 2 milliseconds. This average appears to be stable. Unfortunately, there is a jitter of 1 to 2 milliseconds between the time the samples are taken and the time TENEX increments its internal tables. PFSTAT should probably be modified to subtract sampling time from the figures. For the moment, however, the reader must mentally subtract 2 milliseconds from each incremental runtime. For merged samples, however, a multiple of 2 milliseconds must be subtracted. For example, sample 55 is a merger of 9 samples, so 18 milliseconds must be subtracted instead of 2.

Figure 4 shows a login call on the Works Manager. This is the latest Works Manager, which incorporates version 2 of the Works Manager Table Facility. The results we saw previously were for version 1. As sample 46 demonstrates, the Works Manager spends 11 milliseconds of CPU time waiting for and then receiving the procedure call message. All the times in figure 4 are for this fork only. The 11 milliseconds is a bit puzzling, as the only process which does anything significant during this call is the MSG fork. In any case, the Works Manager then spends 44 minus 18 or 26 milliseconds verifying that this is a valid procedure call message and deciding to call the Login procedure in module WMLOGS (sample 55). The Login routine then spends 14 milliseconds converting the message into internal representation and simultaneously verifying that the message has the correct form for a login request (sample 57). Login next retrieves the node entry corresponding to the project and node name given in the message, consuming 30 milliseconds in the process (sample 59). It appears that a lot of this time is due to the very simple and somewhat inefficient entry name matching algorithm in the Works Manager Table Facility. Next, the Login routine builds an Active User Entry by copying elements from the node entry. The only really expensive operation here is 82 milliseconds to copy the list of tool rights (sample 77). At sample 85, Login takes 11 milliseconds to create the new Active User Entry. Login then puts the updated node entry back into the online database, consuming an alarming 136 milliseconds in the process (sample 91). Login then spends 20 milliseconds checking the User Identification Entry for messages (sample 93). Finally, Login takes 57 milliseconds to format a reply message and send it (sample 97). After Login returns to the Works Manager main program, the main program almost immediately starts to wait for another procedure call message (sample 113).

In summary, Login took a total of 429 milliseconds, or 86 milliseconds more than it did when it was tested in August of 1978 with version 1 of the Works Manager Table Facility. A comparison with the detailed timings for the version 1 test shows that in the new timings the only significant differences were that copying the list of tool rights took 37 more milliseconds and putting the node back took 43 more milliseconds. The extra 37 milliseconds for the copy is easy to explain; the node logged into when testing version 2 had about twice as many tool rights as the one used when testing version 1. The 43 milliseconds for updating the node was much more disturbing, so another test run was made, this time requesting PFSTAT to magnify the call to put the updated node away. Figure 5 shows only that call.

no.	type	lvl	caller PC	called PC	realtm	runtm	dlreal	delrun	dlpage
90	call	3	WMLOGS 41071	OPNUTS 253330	103796	1110	0	0	0
91	call	4	OPNUTS 253356	OLRUTS 261002	103798	1112	2	2	0
92	retn	4	OPNUTS 253356	OLRUTS 261002	103848	1117	50	5	0
93	call	4	OPNUTS 253466	MSKUTS 263533	103861	1120	13	3	1
94	retn	4	OPNUTS 253466	MSKUTS 263533	103863	1122	2	2	0
95	call	4	OPNUTS 253567	WMTUTS 270420	103881	1125	18	3	1
96	retn	4	OPNUTS 253567	WMTUTS 270420	104261	1249	380	124	4
97	call	4	OPNUTS 253636	ENTUTS 256557	104264	1251	3	2	0
98	retn	4	OPNUTS 253636	ENTUTS 256557	104278	1266	14	15	0
99	call	4	OPNUTS 253650	OLRUTS 261215	104280	1268	2	2	0
100	retn	4	OPNUTS 253650	OLRUTS 261215	104281	1269	1	1	0
101	retn	3	WMLOGS 41071	OPNUTS 253330	104281	1269	0	0	0

Fig. 4

The subroutine calls shown in Figure 4 are all in WMTPKG, a set of routines which lie between the Works Manager top level routines on one side and the Works Manager Table Facility and the Information Retrieval System on the other side. The actual call to put the node entry away is made by the subroutine called at sample 97. This call takes only 13 milliseconds. The expensive call is the one at sample 95, which consumes 122 milliseconds. This is a call to a garbage collection routine, and happens to be quite unnecessary. In other words, we have found a performance bug in the Login routine. (This bug has since been fixed.)

A brief explanation is in order of what this garbage collection is all about. As far as the Works Manager Table Facility is concerned, a table entry is just a block of arbitrary numbers. Actually, an entry body is a data structure called a Dynamic Relocatable Table. The use of the word "table" here is unfortunate, and comes in part from the fact that the Works Manager is implemented in layers. In any case, a dynamic relocatable table, here a node entry, will grow in length if nonscalar elements in it are replaced. The only way at present to retrieve the space is to do a garbage collection, which consists of copying, element by element, the entire data structure. This is what is taking 122 milliseconds. Here the garbage collect was totally useless, since the only element changed was a scalar.

This also explains 37 of the 43 extra milliseconds that this garbage collection took with version 2 compared to version 1. A garbage collect is a copy of all elements, including the list of tool rights. As we saw, the longer list in the new node took 37 extra milliseconds to copy.

SUMMARY OF CPU TIME CONSUMED IN WM-LOGIN

1. A total of 429 milliseconds was expended in all.

2. 77 milliseconds, or 18% of the total, was spent retrieving table entries.

3. 111 milliseconds, or 26% of the total, was spent sending, receiving, and decoding messages.

4. 241 milliseconds, or 56% of the total, was spent manipulating table entries. Of this time, 122 milliseconds was due to a bug. Of the remaining 119, at least 82 milliseconds was due to the list of tool rights.

Fig. 5

Figure 5 summarizes what we now know about the Works Manager Login call:

A total of 429 milliseconds of CPU time was consumed in all. Only 77 milliseconds, or 18% of the total, was spent retrieving tables. Thus few of the improvements we suggested earlier for the Works Manager Table Facility would have any noticeable effect, at least for a Login call. The only potential problem is that we might get into trouble retrieving nodes if there got to be a lot of them, so we should probably do something about entry name searches before NSW grows too much larger.

Again, only 111 milliseconds or 26% of the total was spent sending, receiving, and decoding MSG messages. This is a relief, as some of the utility routines which handle MSG messages are large and look threatening. A whopping 241 milliseconds, or 56% of the total, was spent manipulating table entries. Almost half of this time is due to a bug, an unnecessary garbage collection. Furthermore, 69% of the remaining 119 milliseconds was spent copying the list of tool rights.

Thus, if we got rid of the garbage collect and found some way to get rid of the time spent copying the list of tool rights then Login would take about 225 milliseconds -- just over half as much time as it does now.

This is only an example of how we would go about planning for further reductions in Works Manager runtime. Before we actually do any reducing, we should perform the same type of analysis on all major Works Manager procedure calls. We should then use those results, plus the results of BBN's system tests and Manni Chandi's higher level model, to plan which changes will be most cost-effective in terms of overall NSW performance.

Appendix B

Batch Job Package - External Specification

This document is written in the style of Appendix 1 of "The Foreman: Providing the Program Execution Environment for the National Software Works" by R. Schantz and R. Millstein, which is used herein as a reference. It describes functions of the Batch Job Package (which may be invoked by WMO) and a function of WMO (which may optionally be invoked by BJP). All invocation messages are generically addressed; requested replies on the other hand are specifically addressed to the invoking process.

The Batch Job Package (BJP) on a Batch Tool Bearing Host (BTBH) cooperates with Works Manager Operators (WMO's) to control the execution of NSW batch jobs. Once an NSW user has submitted a job, it is the responsibility of a WMO and BJP to execute the job and to produce status reports as required.

WMO serves a role analogous to that of the Foreman in that WMO keeps the Local Name Dictionary (LND) for the job and supervises required file prestaging and delivery operations. BJP includes all functions (exclusive of file transfer and translation) required at BTBH to accomplish batch job execution. Its conversational partners are the WMO's through which jobs are submitted.

BJP's data base is concerned with the management of all NSW jobs in progress at BTBH. Associated with each job is the generic process address of the WMO which submitted it, and is therefore its conversational partner with respect to that job throughout the duration of the job. The following discussion relates to messages for individual jobs and therefore is cast in terms of communication between a single WMO and BJP. It is assumed that the process address of the submitting WMO is recorded in BJP's data base, and is used for addressing messages from BJP. Should any WMO disappear (because of an error) an as yet unspecified restart sequence must be executed.

Job naming considerations

Each WMO maintains a queue of up to 256 jobs in progress. When an NSW user submits a job, the WM to which the user is assigned contacts a WMO with the job request. WMO assigns the job to an unused location in the queue, an assignment which remains intact throughout the stages of job execution.

Every time WMO is "cold-started" all entries in its queue are marked as free. All such cold starts are explicated by maintaining a WMO cycle number which is initially 1 and incremented (except that the successor to 16383 is 1) each time such a cold start occurs. NSW's name for a job is a triple of indices: the WMO host number, that WMO's cycle number, and the position within that cycle's job queue. Every time WMO contacts BJP regarding an NSW job, its NSW name is included in the message.

BJP may wish to create local names for NSW jobs. This optional character string is returned by BJP when a job is accepted. WMO will retain this name in its job queue and supply it along with the NSW name in messages relating to the job.

If BJP can supply cycle and number, local name is optional; if it can supply local name, cycle and number are optional. Complete information is preferred for error checking.

BJP must respond to a small number of messages from WMO, and initiate a message of its own - as follows:

- WMO's initial contact with BJP is to request allocation of a workspace and assignment of local names for job output files. Arguments include batch tool name, cost estimate (in machine dependent units), priority, and a list of size estimates for job output files. Response includes the workspace name (WS, a character string), a status indicating whether the job was accepted or rejected, and a list of local names assigned to output files.

- Once file pre-staging is complete, WMO will request BJP to submit a given local file for batch execution. The response is job status.

- BJP will notify WMO when a job is done. This message includes time and charge information.

- WMO will request BJP to delete a job. BJP's response confirms the deletion and reports any abnormal conditions.

- WMO may inquire as to the status of a job as well as a Boolean which is true if the job is to be continued or false if it is to be cancelled. BJP's response includes a user-oriented character string as well as the job's status.

All job execution sequences begin with an allocation request:

```
BJP-ALLOCATEJOB (tool-id-list, accounting-list, tool-dependent-
                 parameter-list)
    -> tool-id-list, status-list, workspace-descriptor
```

where the parametric data are:

tool-id-list: LIST(cycle-no, tool-instance-id, local-name)

- cycle-no:
  index, with 0 denoting "unknown."
- tool-instance-id:
  integer, with 0 denoting "unknown."
- local-name:
  charstr; if length is 0 then is "unknown."

accounting-list:
see reference pages A1-3 and A1-4.

tool-dependent-parameter-list: LIST(n-charstrs)

status-list: LIST(status-code, qcan-proceed, status-report)

- status-code:
  index =0-> not found
  =1-> allocated
  =2-> scheduled
  =3-> running
  =4-> halted
  =5-> deleted

- qcan-proceed:
  boolean =true -> can proceed
  =false-> cancellation requested

- status-report:
  charstr; if length = 0 then is null report

workspace-descriptor:
see reference page A1-2

Parameter usage

Tool-id-list:

Every message between WMO and BJP includes tool-id-list. This applies to both invocations and replies. When WMO sends a tool-id-list it always includes cycle-no and tool-instance-id, and local-name once WMO learns of it. Most messages from BJP to WMO are specific replies. The returned tool-id-list may include cycle-no and tool-instance-id if desired; WMO will perform consistency checks if they are returned. Should a local name be returned, then it will be recorded and supplied to BJP in all subsequent messages from WMO to BJP. This name must be unique for that BJP. Local-names cannot be reused by a BJP. Different BJPs may use the same name. Should BJP return the local-name once it has been recorded, consistency checks will be made on it, as well.

All other parameters are as described in the reference document, except status-list, which is straightforward.

All subsequent messages from BJP to WMO report on the state of a particular job. BJP's response requirements seem to be as follows: given a message about a job, inform WMO of the new state of the job. This yields a simple response algorithm for BJP, as BJP learns about job completion via polling or a message from BTBH's operating system; and the required response is also a status report to WMO.

```
BJP-QUERY (tool-id-list, qproceed)
      -> tool-id-list, status-list, accounting-list
```

where the additional parameter is

```
qproceed:
    boolean  =true-> can proceed
             =false-> cancellation requested
```

Note that this simplifies WMO processing, as Status is the only variable of interest to it - the remainder of the message data is simply recorded. If Local-name is supplied, then Cycle-no and Job-no are optional, and vice-versa.

BJP-ENDJOB (tool-id-list)
-> tool-id-list, status-list, accounting-list

Note that this is not the same as BJPQUERY(-,F) since it is used by WMO after file delivery and recording of final job charges is complete.

Functions implemented within WMO

WMO-JOBHALTED (tool-id-list, status-list, acounting-list)

This is the only function in WMO invoked by BJP. It requests WMO to initiate job delivery operations. It is an optional feature: alternatively, WMO can issue BJP-QUERY periodically until it finds the status code corresponding to job halted. If implemented, then either local-name or tool-instance-id must be supplied (unless BJP limits itself to a single NSW job at any given time).

Optional functions which can be implemented within any Batch Job Package

BJP-STARTJOB (tool-id-list, workspace-descriptor, filename)
-> tool-id-list, status-list, accounting-list

where the additional parameter is filename:
see reference page A1-2

This function is optional. The named file is taken as the job control statements for the given job. Alternatively, if BJP's file package can initiate job execution when a file of a given kind is created (such as (QF=<jobname>, DC=IN) names in CDC SCOPE systems), BJP-STARTJOB need not be supported.

WMO will be driven by a table taken from the tool descriptor to sequence the procedures which actually get invoked. This allows supporting a variety of batch hosts; e.g. - hosts where filespace must be reserved and those where it doesn't, and hosts which do WMO-JOBHALTED and those that don't.

Transportability

We define transportability as follows: an NSW component/utility is transportable if it can be copied from one host to another and be immediately executed with no patching or modification beyond naming the transported executable file to suit the new system's naming conventions. This definition covers transport between both similar (TENEX - TENEX, TOPS20 - TOPS20) and dissimilar (TENEX-TOPS20-TENEX) systems.

All NSW components and utilities maintained by MCA had to be modified to achieve transportability. We identified two classes of problems to be solved: (1). Differences between TENEX and TOPS20 at the interface to the operating system, i.e. different behavior and/or naming of JSYSes; (2). Tailoring - Configuration - specific data built into components that had to be patched for each NSW system on each NSW host, e.g. the NSW filespace name compiled into the File Package. The first problem we solved by encapsulating most JSYS calls in three JSYS routine packages with system insensitive call interfaces; we solved the second by providing a Global Tailoring File which captures configuration information for each NSW system on each host.

JSYS encapsulation

We encapsulated JSYSes by dividing the JSYSes used throughout our NSW components into three groups, and providing three corresponding packages of BCPL routines to access the JSYSes indirectly. The groups are:

(1). File handling, day/time, and error handling JSYSes

(2). PMAP and related file mapping JSYSes packaged to do page-oriented reading and writing to files.

(3). Utilities to access the ARPANET system tables maintained on both TENEX and TOPS20 hosts.

The call format for all the routines in these packages is consistent and uniform; success or failure of each call is signaled in a uniform way, and a system-produced error message is returned whenever available. Obtaining the latter typically requires two further JSYS calls (GETER and ERSTR) which formerly had to be placed in line; thus much of our code which calls the encapsulations has been reduced in size, and is more readable. When required, checking for TENEX VS. TOPS20 is done at run time. The contents and characteristics of the three encapsulation packages are:

1. Common JSYS package (JSYSUT in BCPPKG

- JSYSes which have different names and arguments on 10X and TOPS20, encapsulated so that arguments produce results identical to execution on TENEX:

STDIR/RCDIR	(directorySTring to directory number)
CNDIR/ACCES	(connect to directory)
GTAD	(get time and day. TOPS20 internal time converted to TENEX)
ODTIM	(output time. TENEX internal converted to TOPS20 if required)

- JSYSes with identical names, but requiring differing defaults or argument formatting to execute identically on TENEX and TOPS20.

GTJFN	(Get JFN. TOPS20 specific code to handle structure references)
PMAP	(Map fork/file page. To handle differences in unmapping)
DELDF	(expunge directory. Different argument sequence)

- JSYSes with identical behavior on both systems: Encapsulation provides consistent defaulting and error handling.

OPENF	(Open file)
CLOSF	(Close file)
CLZFF	(Close fork's files)
RLJFN	(Release JFN)
DELF	(Delete file)
SFPTR	(Set file pointer)

JFNS	(JFN to string)
RNAMF	(Rename file)
SIZEF	(Size of file)
BIN,BOUT	(Byte in, out)
SIN,SOUT	(String in, out)

2. PMAP I/O package (PMPUTS in BCPPKG)

. A group of five routines which use the PMAP JSYS to do page-at-a-time I/O between a file and core memory. Encapsulates the difference beween TENEX and TOPS20 PMAP, and isolates the user from the subtleties of using PMAP to create file pages. The operations supported one:

Getting a file page into core.

Putting a core page into a file.

Locating the next file page.

Getting and modifying some parts of a File Descriptor Block.

3. Host tables package (HOSTUT)

. A group of routines which support the following activities:

One-time read of host tables into core.

Return host (arpanet) number given name.

Return host name and nicknames given number.

Return host operating system type given number.

These packages were thoroughly tested with several straightforward test drivers; all NSW components were than laundered to remove direct in-line JSYS calls. All utilities which did not require the Global Tailoring File facility then became immediately transportable.

Global Tailoring File Facility

This facility consists of the following elements:

(1) The global tailoring file, a fixed format ASCII file encoded so as to be untypeable. This file must exist in the LOGIN directory for each NSW system on each NSW host; it contains the following entries:

NSW filespace name

WMO Job Queue File name

Checkpoint directory name

External process MSG call timeout value

Component logging flag

WMO sleep interval (between each queue in job queue file)

(2) A utility (GLBTAL) which can create or list the contents of this file, and replace individual entries.

(3) A package of utilities (GLBUTS in NSUPKG) which allow components to read entries from the file).

(4) A set of descriptors for the file entries compiled into GLBTAL and all accessors of the file.

Use of this facility requires the following discipline: The Global Tailoring File must exist in each TENEX/TOPS20 NSW LOGIN directory, and its format must be compatible with the compiled-in descriptors for all accessors in that directory. Reformatting the file requires re-compilation, reloading and re-distribution of all accessors. This has not proved burdensome to date, as the file changes format rarely.

In addition to providing configuration information (directory and file names), the facility supports a crude facility for turning test VS. user systems by allowing manipulation of parameters directly related to performance (timeout values, inhibition of logging.

The components and utilities using this facility are:

Works Manager

File Package

Works Manager Operator

Checkpointer

WMO Utility

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.